

**THREE-DIMENSIONAL MODELING SYSTEMS:
CONFORMITY AND USABILITY AMOUNG ALICE
AND RAY DREAM STUDIO 5**

A Thesis in
TCC 402

Presented to

The Faculty of the
School of Engineering and Applied Science
University of Virginia

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Science in Engineering Science

Submitted by

Riesa Susan de Beer

May 8, 1998

On my honor as a University student, on this assignment I have neither given nor received unauthorized aid as defined by the Honor Guidelines for Papers in TCC Courses

Signed _____

Technical Advisor _____ Date _____

TCC Advisor Approval _____ Date _____

i. Abstract

This project addresses the lack of conformity among modeling and animation packages available in today's market. To accomplish this, an animation sequence was devised to form a basis of comparison across the modeling packages. This project focuses its attention on two software packages, Alice and Ray Dream Studio 5.

Alice is not a modeling application. Rather it is a scripting environment that allows for the construction of 3D scenes. Ray Dream Studio 5 is a powerful, three-dimensional modeling package. This package allows an animator to create 3D illustrations and animations. These two packages are representative of the diversity among modeling and animation packages.

During the course of this project a few bugs were discovered within the Alice software. The biggest of these errors is Alice's memory usage. It appears that Alice does not correctly reset portions of its available memory. This alters the performance of the scripts and eventually causes Alice to have a memory fault and shut down. The second bug is related to Alice's method of playing sounds. Alice does not wait for the completion of a sound request nor can a request be stopped once it has been initiated. Recommendations were also made to improve Alice's usability. Ray Dream did not have any apparent bugs. Ray Dream's biggest flaw is not accommodating for the needs of novice animators. Suggestions were made for meeting beginners' demands.

Both Alice and Ray Dream Studio 5 can greatly improve their usability by not targeting a specific user level. To be truly functional, a modeling package

should nurture an animator's needs and provide support for the growth of those needs.

ii. Dedication

For my wonderful parents,
who have applauded my success, supported my goals, and accepted my failures.

I love you.

Table of Contents

I. ABSTRACT	1
II. DEDICATION.....	3
III. LIST OF FIGURES.....	6
IV. FOREWORD.....	7
1. INTRODUCTION	11
1.1 PROBLEM DEFINITION	11
1.2 MODELING SYSTEMS.....	12
1.2.1 Alice (Beta Version).....	13
1.2.2 Ray Dream Studio 5.....	14
1.3 THE ANIMATION SEQUENCE.....	14
1.4 RATIONALE AND SCOPE	15
1.5 OUTLINE OF REPORT	16
2. ANIMATION AND MODELING CONCEPTS	18
2.1 PROLOGUE.....	18
2.2 KEYFRAMES.....	18
2.3 MODELING PROGRAMS	19
2.4 TEXTURE IN 3D MODELING	21
2.5 TIMING IN 3D MODELING.....	21
2.6 SCRIPTING LANGUAGES	22
2.7 LUXO JR. AND THE ANIMATION COMMUNITY.....	23
3. ALICE.....	25
3.1 THE TELEPHONE SEQUENCE.....	25

3.1.1 <i>Difficulties Encountered</i>	25
3.2 THE ANIMATED LAWN TOUR	27
3.2.1 <i>Difficulties Encountered</i>	28
3.3 THE SOFTWARE CAPABILITIES	31
3.3.1 <i>Requirements</i>	31
3.3.2 <i>The Graphical User Interface</i>	31
3.3.3 <i>Animation Techniques</i>	33
4. RAY DREAM STUDIO 5.....	34
4.1 THE TELEPHONE ANIMATION SEQUENCE	35
4.1.1 <i>Difficulties Encountered</i>	35
4.2 THE SOFTWARE CAPABILITIES	37
4.2.1 <i>Requirements</i>	38
4.2.2 <i>The Graphical User Interface</i>	38
4.2.3 <i>Modeling Techniques</i>	41
4.2.4 <i>Animation Techniques</i>	42
5. RECOMMENDATIONS AND CONCLUSIONS.....	46
5.1 ALICE.....	46
5.2 RAY DREAM STUDIO 5.....	48
5.3 SUMMARY.....	50
6. REFERENCES	51
APPENDICES.....	53
APPENDIX A: ALICE BETA QUICK REFERENCE CARD	54
APPENDIX B: ALICE CODE FOR TELEPHONE ANIMATION SEQUENCE.....	56
APPENDIX C: ALICE CODE FOR THE LAWN TOUR	57

iii. List of Figures

FIGURE 1 - USE OF A MODELING SYSTEM. SOURCE: [MARTIN, 1997]	12
FIGURE 2 - LUXO JR. AND BALL SOURCE: [PIXAR, ONLINE]	23
FIGURE 3 – THE “CHARMED” TELEPHONE IN ALICE	26
FIGURE 4 – UVA LAWN TOUR IN ALICE	28
FIGURE 5 - THE SCRIPT EDITOR	32
FIGURE 6 - THE CAMERA	32
FIGURE 7 – THE “CHARMED” TELEPHONE IN RAY DREAM STUDIO 5	34
FIGURE 8 - VIEW OF TELEPHONE CORD IN THE SCENE	36
FIGURE 9 - VIEW OF TELEPHONE CORD IN THE MODELER	37
FIGURE 10 - THE PERSPECTIVE WINDOW	38
FIGURE 11 - THE TIME LINE WINDOW	39
FIGURE 12 - THE BROWSER PALETTE	40
FIGURE 13 - THE PROPERTIES WINDOW	40

iv. Foreword

Animation has always interested me. When I approached Professor Worthy Martin regarding a thesis topic, one of his suggestions included three-dimensional modeling packages. I jumped at the idea, even though I knew very little regarding animation or modeling. Since then I have learned an invaluable amount. I greatly appreciate the opportunity to have worked on such a fascinating topic. I would like to give special thanks to Professor Martin who aided in the development of this thesis topic and who was particularly helpful in guiding me through unfamiliar territory. I would also like to thank Steve Audia for his endless help regarding Alice. It was a pleasure to work with him. I would also like to thank Professor T. E. Hutchinson for granting me access to his systems laboratory. I realize that computer time is a scarce resource and I greatly appreciated it. Also, I would like to thank all of those who supported me, helped me, and pointed me in the right direction during the course of this project.

v. Glossary

Alice	Authoring system for 3D graphics. This software package runs in a Windows 95/NT environment.
Animation	A simulation of movement created by displaying a series of pictures, or frames. Cartoons on television, for example, is one example of animation.
Bitmap	A pixel-based image
Complex Object	An object constructed of several simple objects that are linked or grouped. For example, a telephone is comprised of a cord, a receiver, buttons, and a base.
Dynamic module	An animation sequence in which the animation responds to user input.
Extruding	Giving a 2D object a dimension of depth, transforming it into a 3D object
Flip book	A method of viewing animation by first saving the images in RAM and then viewing them in sequential real-time
Frame	A single image in a sequence of images.
Frame rate	Number of frames displayed per second.
Free form modeling	A method of modeling that created 3D objects through sweep paths and cross sections.
Hot Spot	A special, active point inside or in the surface of an object. It is used as reference for the object's location and serves as the center of weight. All motion revolves around this point.
Keyframe	A moment, or frame, in the animation sequence where the object is at a characteristic or extreme location. A keyframe is also the source and target for tweening.
Mesh form modeling	A form of modeling that allows the animator to create a 3D object by directly manipulating its surface.
Modeling	The process of representing 3-dimensional objects in a computer.

Modeling Software	The category of software that represents 3-dimensional objects on a computer. This includes CAD/CAM and animation packages.
Motion Path	A curve that shows the path an animation will follow during the course of an animation. It is relative to the object, not the viewpoint.
Object	Any 3D volume or other item that appears in the world, including the camera and lights.
Object hierarchy	The relationship that exists between an object and its parts. Parts in low levels are dependent on parts in higher level.
Object motion	The motion performed by an objects during the course of an animation.
Pan	To rotate a camera's position around its vertical axis.
Pixel	An acronym for picture element. One dot in a 2D image. Computer images are generated as an array of such dots, each having a specific color
Point of View	The position and angle from which a scene is viewed.
Python	An interpreted, interactive, object-oriented programming language. The scripting language for Alice.
Ray Dream Studio 5	A powerful, 3D modeling package that allows an animator to create 3D illustrations and animations.
Ray Tracing	Rendering technique that works by simulating the path of a single light ray as it would be absorbed or reflected by various objects in the image. To work properly, the artist must specify parameters of the light source (intensity, color, etc.) as well as all the objects (how reflective or absorbent the materials are).
Real-time	The viewing of an event as it is happening.
Render	The process of adding realism to a computer graphics by

adding three-dimensional qualities such as shadows and variations in color and shade. One technique for rendering graphics is called ray tracing.

Scripting language	The underlying language, supported by a modeling package, that allows an animator to write sequences of code in order to animate objects and scenes.
Scientific visualization	Use of animation in the sciences. Advanced computer software has been used in the simulation of military exercises, weather patterns, and even biological processes.
Single frame animation	The process of creating animation one frame at a time, as opposed to seeing the animation run as it is intended to.
Static module	An animation sequence in which the animation responds to user input.
Stereo pairs	A method that allows an individual to see a 2D object in three dimensions. For example, the red-blue glasses provided in cinemas.
Sweep path	The curve or line along which the shapes are extruded when modeling.
Texture	In 3D graphics, the digital representation of the surface of an object. In addition to two-dimensional qualities, such as color and brightness, a texture is also encoded with three-dimensional properties, such as how transparent and reflective the object is.
Texture Mapping	Wrapping a texture around any 3D object, once the texture has been defined.
Tilt	To rotate a camera's view upward or downward on its horizontal axis.
Tweening	Creating in-between images based on a beginning and ending keyframe.

1. Introduction

Recently, George Lucas re-released the Star Wars Trilogy. To entice fans to see the familiar movies again, he included scenes that had never aired before. The reason these scenes were originally left out was lack of appropriate technology. Today's camera tricks made it possible for Jabba to leave his palace and pay Han a visit. These "camera tricks" are more commonly known as three-dimensional modeling packages. The three-dimensional modeling packages available today do not lack in performance, but conformity. When designing modeling packages, designers did not have a "recipe" to follow. Consequently, the learning curve of adapting to various modeling packages is very steep. This problem leads to the question, what animation techniques should a 3D modeling package provide for a user?

The purpose of this project is to address this apparent divergence within modeling and animating software. One direction that this project has taken is the design and implementation of an animation sequence in various modeling and animating environments. Ray Dream Studio 5 and Alice were chosen as the modeling and animating environments. The animation sequence will form the basis of comparison between Alice and Ray Dream. The sequence is intended to highlight the various strengths and weaknesses of these packages.

1.1 Problem Definition

When most people think of animation, providing motion to an inanimate object is most often what comes to mind. Animation also includes time-varying position, shape, structure, color, texture, and lighting of an object [Foley *et al*,

1990]. Another common misconception is that computers animate. Computers only provide a vehicle through which people animate. For this reason, understanding how a 3D modeling package acts as an interface between the animator and the computer is important.

Many 3D modeling packages are commercially available. Unlike most software that belong to the same genre, there is little conformity among modeling packages. There is also little uniformity concerning the implementation of animation techniques, such as texture mapping and object hierarchies. In some cases a modeling package will even contain an “embedded animation language so that the simulation and animation processes are simultaneous” [Foley *et al*, 1990]. Consequently, the learning curve of adapting to various modeling packages is very steep. Is it possible to provide a general “recipe” for a three-dimensional modeling package without sacrificing usability and performance?

1.2 Modeling Systems

Modeling systems and computer animation have applications in numerous areas. The most predominant categories are: science, entertainment, research, advertising, education, and simulation. Computers do not animate, people do. Computers only provide a vehicle through which people animate. For this reason, understanding how a three-dimensional modeling package acts as an interface between the animator and the computer is important.

From a designer’s point of view, a modeling package can be used as a tool to create an instructional module. For example, a teacher may create an instructional module used by students in his or her class. This relationship is

demonstrated in Figure 1. This instructional module can be static or dynamic. A static module does not respond to the user whereas a dynamic module responds to input from its user. A designer focuses on how the modeling package allows a user, or animator, to create such a module. What tools

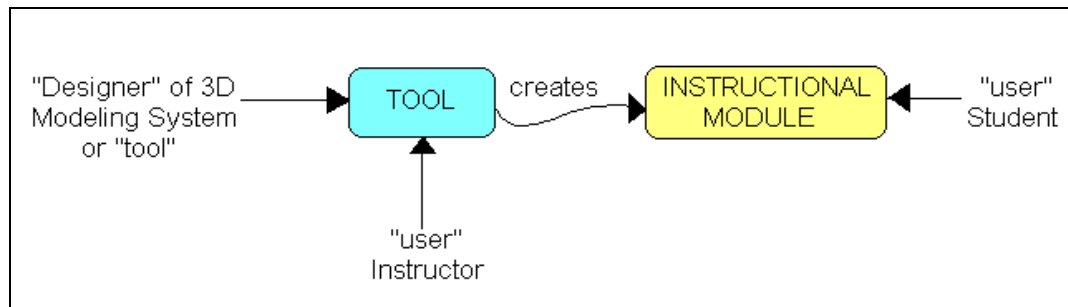


Figure 1. Use of a Modeling System. Source: [Martin, 1997]

and applications should be provided to the “instructor?” The 3D modeling package should allow for ease in specifying behavior, object motion, and articulated motion. The package should also allow for ease of coordinating multiple images on screen and texture mapping.

1.2.1 Alice (Beta Version)

One of the software packages included in this study is Alice, an interactive 3D graphics environment. Alice is not a modeling application. Rather it is “primarily a scripting and prototyping environment for 3D behavior” [Alice, online]. This shortcoming, however, does not exclude the Alice software from this study. Python is the scripting language that Alice supports. Alice is capable of reading common 3D file formats. This enables an animator to model an object with another modeling package or obtain a 3D object from the World Wide Web and insert the object into Alice. Objects in Alice can “move, spin, change color, make sounds, react to the mouse and keyboard, and more.” Alice comes with a

development environment and a collection of three-dimensional objects.

1.2.2 Ray Dream Studio 5

Ray Dream Studio 5 is a powerful, three-dimensional modeling package. This package allows an animator to create 3D illustrations and animations. Ray Dream Studio's animation technique is key-frame and timeline driven. Unlike Alice, it does not require scripting language to animate worlds. This modeling package also provides the user with many pre-modeled objects. Ray Dream Studio 5 provides the animator with two forms of object modeling. The first is free form modeling. This method of modeling uses cross sections, sweep paths, and scaling envelopes to create 3D objects. The second method is a mesh form modeling. This method allows the animator to directly manipulate the surfaces of three-dimensional objects.

1.3 The Animation Sequence

An animation sequence was devised to form the basis of comparison between the aforementioned modeling packages. The sequence was based on an Indian snake charmer. In this sequence the snake will be replaced by a telephone and the computer will act as the charmer.

The opening scene contains a phone sitting on a desk. A tune will begin to play in the background. As the music grows stronger, the telephone begins to respond. The phone raises its receiver and moves it like a snake's head. The "charmed" telephone dances to the enchanting melody. If the music is interrupted, the receiver will drop. Otherwise, the receiver will finish its "dance" and come to rest on the phone base. This idea was inspired by Pixar's short animated film,

Luxo Jr. Later, this idea was modified to have the phone respond to a mouse button click instead of a tune.

Another aspect of this project was to create a virtual tour of UVA's Lawn. This is an example of a static instructional module. The Lawn is built by Steve Audia in a different modeling system. The user is allowed to follow the tour guide around the Lawn and inspect all the buildings. Despite this interaction, the Lawn tour is not dynamic. For example, the door of a building will not open as the user approaches. For the purpose of designing a tour, a static model is sufficient. The user of the virtual tour is interested in historical facts and architectural details, not visual interaction.

1.4 Rationale and Scope

This project may have an impact on the animation community. A general consensus on the mechanisms provided by the software will ease the animation burden for users of three-dimensional modeling systems. The various modeling packages currently available employ a wide variety of techniques. Familiarity with one package's conventions does not provide familiarity with any other packages. Even upgraded versions of the same modeling software have a steep learning curve [Audia, 1997]. This lack of portability is a direct result of the various design choices employed by software designers. No uniform standard for animation techniques exists. A standard method of implementation is also lacking. Providing software developers with a basic set of animation techniques that a three-dimensional modeling package should provide for its user can serve as a casual guideline for them to follow in the design of modeling packages. The virtual

tour of the University of Virginia's Lawn will directly benefit students and prospective students. The tour will be made publicly available via the World Wide Web. From here, any person interested in gaining information on our University's rich history can gain access to the site. The University of Virginia will also benefit from the exposure. Many prospective students are swayed to attend this establishment by the inspiring grounds. A virtual tour will allow prospective students a chance to visit UVA without making travel arrangements.

1.5 Outline of Report

This report contains a literature review, two main chapters, and a conclusion and recommendations chapter. The literature review is intended to familiarize the reader with general animation concepts. The third chapter discusses the animation sequences implemented on the Alice software. The Alice software will be discussed in detail, along with complications involved in implementing the animation sequences. The fourth chapter will discuss the Ray Dream Studio 5 modeling package in a similar fashion. The chapter will focus the software's capabilities and intricacies met while implementing the telephone animation sequence in this modeling platform. In the last chapter a final comparison is drawn between the two software packages, conclusions are formed, and recommendations made for modeling packages. The code for the animation sequences in Alice is presented in the appendices.

2. Animation and Modeling Concepts

2.1 Prologue

From the earliest days, humans have used story telling for communication. Animation first appeared as a flip book in the late eighteenth century. These pictures aided the story teller, allowing him to display an event. Animation developed beyond the flip book to more non-traditional means. For example, stereo pairs give the viewer the impression that he/she is seeing a three-dimensional image when, he/she is actually viewing two appropriately placed flat images [Martin, 1997]. A modern day example are the 3D posters that have become so popular. Upon close inspection, the poster is nothing more than a few hundred brightly colored dots. If you stare at the poster long enough for your eyes to lose focus, and “the strong impression of a three-dimensional object” begins to form in front of the poster [Martin, 1998]. All of a sudden, a collection of dots has transformed into Lady Liberty. Animation capabilities have expanded to include other areas such as “morphing.” Traditional has a long history, and all its aspects are not discussed here. The discussion is limited to the modern adaptation of the century old flip book, keyframe animation.

2.2 Keyframes

To create motion in animation, a series of images is successively drawn at high speeds. Each item in an animation series is called a frame. Frames are the computer equivalent of pages in a flip book. Keyframes are frames in which the object being animated is at characteristic or extreme positions [Foley *et al*, 1990].

For example, keyframes include drawing the first and the last location of an object on screen. After creating the keyframes, the in-between frames or “blank pages” are rendered implicitly. The process of filling the gaps between keyframes is known as “tweening” [MacNicol, 1992]. In traditional, hand drawn animation, this chore fell on the shoulders of an assistant animator. Today, however, most modeling systems perform this tedious task. The task of animating an object approaching from a distance requires only two keyframes. The animator specifies the starting and ending locations as well as the time for the object to approach the viewer [MacNicol, 1992]. Many modeling packages allow for time specification through a time line.

Another common feature is path specification. This innovation allows the animator to specify the path along which the object must travel. The user notes the start and finish time and the computer interpolates the object’s motion in the given time and path constraint. Keyframe animation with a computer not only allows for the timing and positioning of objects, but lighting characteristics as well. The animator can change not only the location of the light source during and animation sequence, but the type of light source as well [MacNicol, 1992]. Using the previous example, a car can now approach from a distance, while the sun rises casting different shadows. Keyframing forms the foundation in modeling programs. Advanced techniques, such as modeling, add to the power of keyframe-based animation.

2.3 Modeling Programs

It is now appropriate to ask: “What is a modeling program?” A three-

dimensional modeling package is a software package that allows a user to generate three-dimensional objects. A modeling package serves as an interface between the animator and the computer hardware. It provides the user with a window in which to create a model of the object and its environment. This interface is then linked to the hardware via a scripting language. According to V.R. Auzenne, “modeling is the creation of the 3D database which serves as the ‘world’ to be portrayed in a synthetic computer graphics sequence.” Models can be classified as either two- or three-dimensional. Even though the image presented on the monitor is two-dimensional, the model used to create the image is three-dimensional. In her book, *The Visualization Quest: A History of Computer Animation*, V.R. Auzenne claims that “three-dimensional animation consists of three principle functions: object modeling, motion specification, and synchronization and image rendering.”

One form of object modeling builds complex models using wire-frame or solid models. Animating a complex object, such as a human being, can be simplified by dividing the object into components [Foley *et al*, 1990]. For example, a human can be divided into an upper body and a lower body. The upper body can be subdivided into a right and left arm. Each arm can consequently be broken down into a lower arm and upper arm, with the lower arm consisting of a hand and fingers. If an animator wants to animate a man raising his hand, great care must be taken to synchronize the arm’s components. For example, for a man waving, having the man’s lower arm move while his hand remained stationary would not be desirable. Object hierarchies ease the difficulty of animating complex objects, by enabling the animator to move a man’s arm, by simply issuing the

statement, “raise arm.” The hierarchical dependence that exists between the components of the man’s arm would ensure that all the components move in unison [Martin, 1997]. In other words, the man’s upper and lower arm will only move if the arm moves. The hand will move if the lower arm moves, and finally the fingers will move if the hand moves.

2.4 Texture in 3D Modeling

Creating a realistic three-dimensional object requires more than accurate structure and motion. Surface texture is an important component in creating realistic 3D models. Textures have complex coloring patterns. One can simulate these patterns by breaking the surface down into large numbers of polygons. Each polygon will have a unique coloring scheme to create the illusion of texture. This approach is very time-consuming and error-prone. A simpler method would be to prerender a graphic texture and store it as a bitmap, a graphic file. The bitmap can then be “wrapped” onto the surface of a polygon, taking into account its surface attributes. This technique is known as texture mapping [Bradford, 1995]. Mapping a rectangular bitmap onto an arbitrary polygon requires only a portion of the bitmap. Otherwise, the bitmap can either match the shape of the polygon or have the same number of vertices. A list of texture coordinates contains the pertinent information regarding the shape of the polygon. A coordinate in the bitmap maps to a vertex in the polygon. The coordinate-vertex assignment is performed by the modeling system as a standard feature [Bradford, 1995].

2.5 Timing in 3D Modeling

One aspect of creating believable animation is timing. A monster racing down the

hall at the “terrifying” pace of one pixel per hour does not have the desired effect. The frame rate is the rate at which each new frame is presented. If the frame rate is adequate, the viewer’s mind will fuse together all the images to create a single fluid impression. Typically frame rates in the range of 15 to 30 frames per second are sufficient. If an animation series has a frame rate lower than 10 frames per second, the image will appear jumpy [Bradford, 1995].

A common trade-off in modeling systems is effective-frame rate versus accurate animation. A complex object requires more time to compute changes in its details to show a accurate rendition of the complex object. In other words, a complex object has a lower frame rate associated with it than a simple object. The reason behind this observation is that a simple object has less associated information to display or update in each frame. However, a monster tearing down the hall without any texture is also undesirable. A compromise must be found between detail and speed.

2.6 Scripting Languages

Often three-dimensional modeling systems make use of a scripting language to support animations. A scripting language is a simple programming language that allows a user to write a script, a list of commands that can be executed without user interaction [PCWebopedia, 1997]. The power of a scripting language lies in their portability and ability to “accomplish remarkable result in only a few lines” [Laird *et al*, 1997]. Modern scripting languages are very expressive and are written to be extended. Extending a language entails binding it together with “external functionality that is not part of the core language”[Laird *et*

al, 1997]. Scripting languages' extensibility makes them seductive for supporting animation modeling systems. Python has a "strong model of object-oriented programming" and is considered one of the "big three" modern scripting languages today [Laird *et al* 1997]. Python is suitable for programs written by many programmers. The language is clean and easy to read. Python has a reputation for being very readable despite many authors and the size of the code involved. Alice uses Python to support its animations.

2.7 Luxo Jr. and the Animation Community

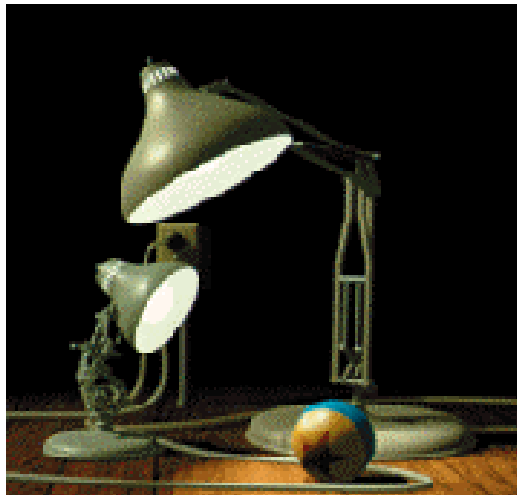


Figure 2 - Luxo Jr. and Ball Source: [Pixar, Online]

In 1986, PIXAR, a graphics company that was originally part of LucasFilm, animated a short film called *Luxo Jr.* This short film was one of the first computer animations to be nominated for an Academy Award [Auzenne, 1994]. The plot is simple. Two desk lamps jump around a desktop to explore their environment. Specifically, the "young" lamp plays with a ball, as shown in Figure 2. Luxo Jr. bends over, pushes the ball, and then watches the ball roll away. As the ball rolls away, it is illuminated by the lamp's light. This film was one of the first sequences

to illustrate the power of computer graphics. It presents itself as a case study. This short film combines all the previously mentioned techniques to give life to a small desk lamp. Even though the software on which *Luxo Jr.* was rendered predates the current software, it is still an excellent example of how to use animation mechanisms.

Within the animation community, a general consensus that the above techniques are fundamental to creating three-dimensional graphics exists. Currently there are no set standards of how a modeling package should implement these techniques. Among modeling systems, there is much variation in animation mechanisms that allow behavior specification. Conformity regarding these issues will not only simplify using a modeling system, but simplify designing one as well.

3. Alice

Alice is a virtual reality engine that allows users to compose and manipulate three-dimensional environments. Alice does not, however, allow a user to model objects. This software package is “useful for describing 3D object behavior, not for creating objects themselves” [Alice: FAQ, Online]. To compensate for this shortcoming, Alice provides its users with hundreds of premodeled objects. Alice also supports common 3D file formats such as .DXF, .OBJ, .X, and .A3D. This allows a user to import an object modeled with other modeling programs or downloaded from the World Wide Web.

3.1 The Telephone Sequence

The animation sequence incorporating the telephone, shown in Figure 3 on the following page, imitates an Indian snake charmer with a technological twist. The snake is portrayed by a telephone and the computer plays the role of the enchanter. Despite the computer's pivotal role, it is not considered a character in the animation sequence. This animation sequence responds to input. The original input was a melody, but later the input was changed to a mouse-button click.

3.1.1 Difficulties Encountered

The Alice software is in its Beta release phase. Therefore it is reasonable for the software to contain bugs that still require fixing. Most of the obstacles encountered stem from idiosyncrasies in the software. The first problem stumbled upon in the Alice software is its method of playing sounds. Like most applications,

Alice takes control of the computer's sound system. Once Alice

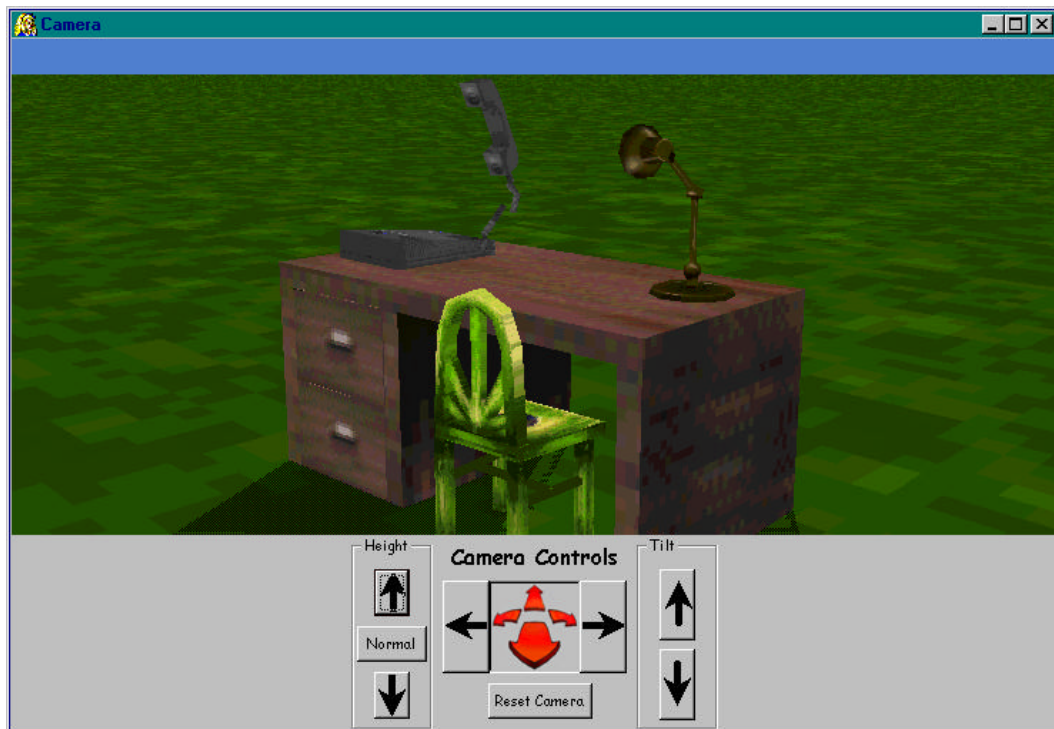


Figure 3 – The “Charmed” Telephone in Alice

requests that a sound file be played, the request cannot be stopped. For example, Alice executes a program that produces a sound. The sound system receives the request and begins to play the associated sound file. If the program is terminated at this point, the sound system continues to play the sound file. This flaw prevents the telephone from responding to the music as originally intended. Instead, the phone was programmed to respond to a mouse-button click.

A second shortcoming in the software is Alice's method of specifying complex object motion. Undue caution was involved in animating the telephone cord. While the telephone base remains stationary, the receiver moves through the air. Nonetheless, it is the cord that breathes life into the charmed phone. The cord sways, straightens, and curls as the receiver dances through the air. To give the

cord these lifelike qualities required a very complicated object hierarchy. Recall that complex objects can be divided into a collection of simpler objects. For example, a telephone consists of a base, a cord, a receiver, and buttons. Intuitively, it makes sense to model the cord as a single unit. This is not wise given Alice's range of motion commands. Alice does not allow for complex object motions such as twisting or stretching. For this reason, the cord must be modeled as a collection of separate links. This makes working with the cord very precarious. It is quite a daunting task to keep all of the links in order and in line. Moving each link individually is tedious and results in very repetitive code that tends to be error prone.

Another problem was encountered concerning Alice's memory use. Upon each performance of the script, the quality deteriorates. This problem was more evident in the animated Lawn Tour. The memory quirk caused the telephone cord to change its behavior during implementation. It was very difficult to determine if the performance of the cord stemmed from the code or the bug. This quirk was compensated for by restarting Alice frequently.

3.2 The Animated Lawn Tour

The animated Lawn Tour is implemented in Alice is intended to imitate a UVA Lawn tour. A purple dinosaur acts as a personal tour guide through an animated version of Thomas Jefferson's academical village. As the dinosaur wanders through the grounds, he stops at the noteworthy sites and relinquishes interesting facts about the buildings. The user has the option of ending the tour at any point by killing the tour guide with a simple button click. This rapid end is a

little brutish, but very effective. The tour progresses in serial order. The only influence the user has on the animation sequence is the untimely demise of the

tour guide. A snapshot of the Lawn tour is shown below in Figure 4.

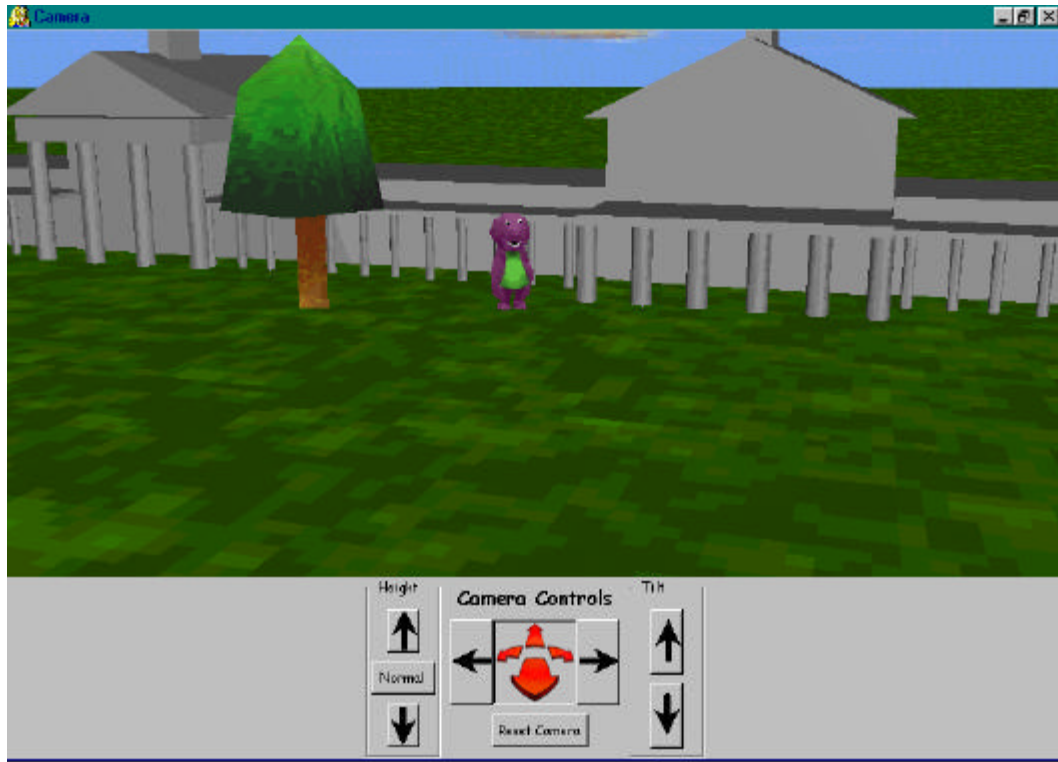


Figure 4 – UVA Lawn Tour in Alice3.2.1 Difficulties Encountered

3.3 Difficulties Encountered

Animation eccentricities in Alice spawned most of the problems met while implementing the Lawn Tour. Alice's method playing sounds was once again a problem. As stated before, if the animation sequence is terminated, the sound file continues to play. In the telephone animation sequence, this quirk created a problem because a sound file could not be interrupted. This obstacle did not pose much of a problem in the Lawn Tour, but, it did make timing very precarious. When Alice requests a sound file to be played, it sends the request, but does not

wait for the completion of the request. While the sound file is playing, the program continues to execute subsequent commands even though the preceding command has not finished executing. In other words, while the dinosaur is explaining the history of the Rotunda, he starts walking to the next Pavilion while still discussing the Rotunda. This is undesirable. To work around this flaw, each sound request is coupled with a wait function. This forces that computer to wait before executing the commands following the sound request. This hack works well, but riddles the code with unnecessary wait functions. The sound system's quirks also complicates changing the sound files. New timing data must be obtained for each sound file.

Alice's looping construct highlighted another bug in the software. The Alice software allows the user to use a looping command to repeat actions. This construct was used to move the dinosaur between the Pavilions and the Rotunda. A walk command and a go forward command were looped together. The walk command moves the dinosaur's legs to create the illusion that he is actually walking. The go forward command moves the dinosaur forward. The interaction of these commands can be seen in the code for the Lawn Tour, exhibited in Appendix D. The looping commands do not produce the same result upon each compilation of the program. The performance of the looping construct deteriorates with each execution. Initially, the looped commands move the dinosaur forward a preset distance. After a couple compilations, the dinosaur seems to hit a glass wall. The dinosaur does not move the same distance as before, but its legs keep moving. Eventually, Alice will have a memory fault and shut

down. It appears that Alice does not correctly reset portions of its memory upon each performance of the script. This is like wiping a chalk board with a dirty eraser. For a while the writing is legible, but it becomes worse each time the board is wiped clean. Eventually it becomes impossible to read the board and it must be wiped down with a wet cloth. During the implementation of the Lawn Tour, Alice was restarted every seven to ten compilations. If the animator were not careful to save frequently, all recent changes made to the script were lost. This is infuriating and time consuming.

In working around the memory bug, another quirk in the Alice software was discovered. Alice provides the user with the ability to place object relative to other objects. Instead of using trial and error to move the dinosaur to the front of the Rotunda, the following command can be used:

```
PurpleDinosaur.place(infrontof, lawn3.Rotunda)
```

Contrary to intuition, this command places the dinosaur in the air above the Rotunda. This problem stems from the location of each object's axis relative to the scene's axis. In an animated world, the scene has an axis that denotes the center of that universe. All objects in the scene are placed relative to the scene's axis. Each object also has its own axis denoting its center. This allows objects to be placed relative to each other using transformations between the axes involved. As the models were created by Steve Audia, the scene's axis is centered on the ground, but the Lawn object's axis is located in the center of the object. This placement is logical when modeling objects, but creates a problem when positions relative to the Lawn are defined. Because of the unusual location of each

Pavilion's axis, the place command results in object placement contrary to intuition. Eventually the dinosaur was moved around the Lawn using the MoveTo command, see Appendix D. It appears as follows:

```
PurpleDinosaur.MoveTo(x, y, z)
```

where x, y, and z are coordinates relative to the scene's axis. The location of the appropriate coordinates were determined by trail and error. This process was very laborious.

3.3 The Software Capabilities

The original idea behind Alice was to create a virtual reality engine simple enough for anybody to use. This requirement precludes Alice from incorporating high-end animation mechanisms such as deformers, but, this software package's strength lies in its simplicity. Many users complement Alice on its ease of use [Alice, online]. The Alice software definitely bides by the adage, less is more.

3.3.1 Requirements

Minimum system requirements are:

- Pentium 90 MHz (120 MHz or faster recommended).
- 40 MB available hard drive space.
- Windows 95.
- Video card with 1MB VRAM (4MB recommended).
- 16 MB RAM.
- CD-ROM drive.

3.3.2 The Graphical User Interface

An Alice world consists of an opening scene, shown in Figure 3 and Figure 4, and a script editor, shown in Figure 5. The opening scene window displays the position of all objects before any animation has begun. This is where the animator arranges the objects. At this point, the stage is set, the actors are ready, but there

is no script to perform [Alice Manual, 1997]. The script editor

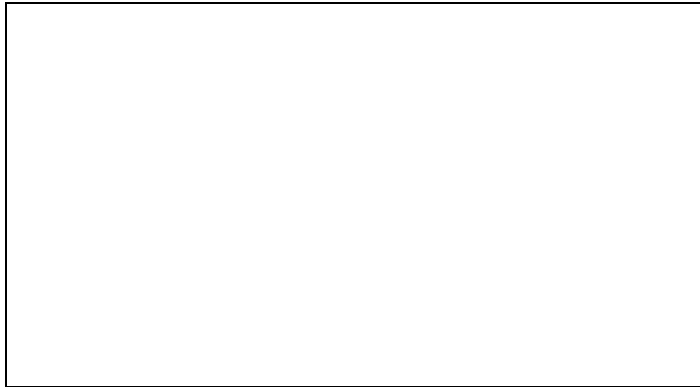


Figure 5 - The Script Editor

allows the animator to “write the play.” The animator can write and execute the script from this window. Each time the script is performed, the opening scene is reloaded. Thus, changing the opening scene changes the objects’ initial positions, but not their action. Changing the script changes the objects’ actions, but not their initial positions. The script editor also comes equipped with a camera window. The camera, shown in Figure 6, allows the user to view the

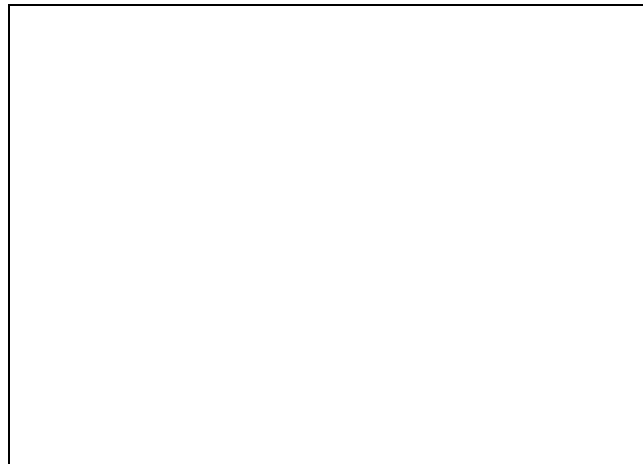


Figure 6 - The Camera

effects of the code. Directions in an Alice world are defined relative to an object. Each scene begins with a camera and a light. The camera provides the user with his or her point of view. The light illuminates objects to make them visible and cast

shadows.

3.3.3 Animation Techniques

The scripting language supported by Alice is Python. Alice prides itself on the simplicity of these commands. For example, to move an object forward two inches, the following command is written:

```
object.move(forward, 2)
```

Ease in defining simple object motion is one of Alice's many successes. Defining commands, such as playing sound, casting a shadow, and setting texture, are all very straightforward. If Alice does not contain the desired command, creating one is just as easy. Alice allows a user to group commands under a heading. For example, the command "dance" can be a combination of simpler commands like spin, move.forward, and roll. The keywords, DoInOrder and DoTogether allow for simultaneous or sequential execution of grouped commands. Now an object can dance when the animator issues the command: dance(). It is really that simple.

The camera and light source are also animatable objects. The camera can move around, pan, and tilt. The light can also be moved, casting different shadows. This can create the illusion of passing time. The Alice Beta Quick Reference Card is provided in Appendix B for a more detailed overview of available commands.

4. Ray Dream Studio 5

Ray Dream Studio 5 is a powerful three-dimensional modeling system. It provides its user with two basic methods of modeling. The animator can create complex 3D objects by combining primitives, basic geometric shapes, or using a mesh or free form modeler. Ray Dream also provides the user with hundreds of premodeled “smart” objects. These objects consist of complex object hierarchies and the user is capable of editing the objects at will. Ray Dream provides the user with a wide variety of high-end animation techniques. Some of these techniques include a time line, keyframe events, and deformers. Ray Dream’s scene wizard guides a user through creating a scene by adding objects, adding lights, and placing the camera.

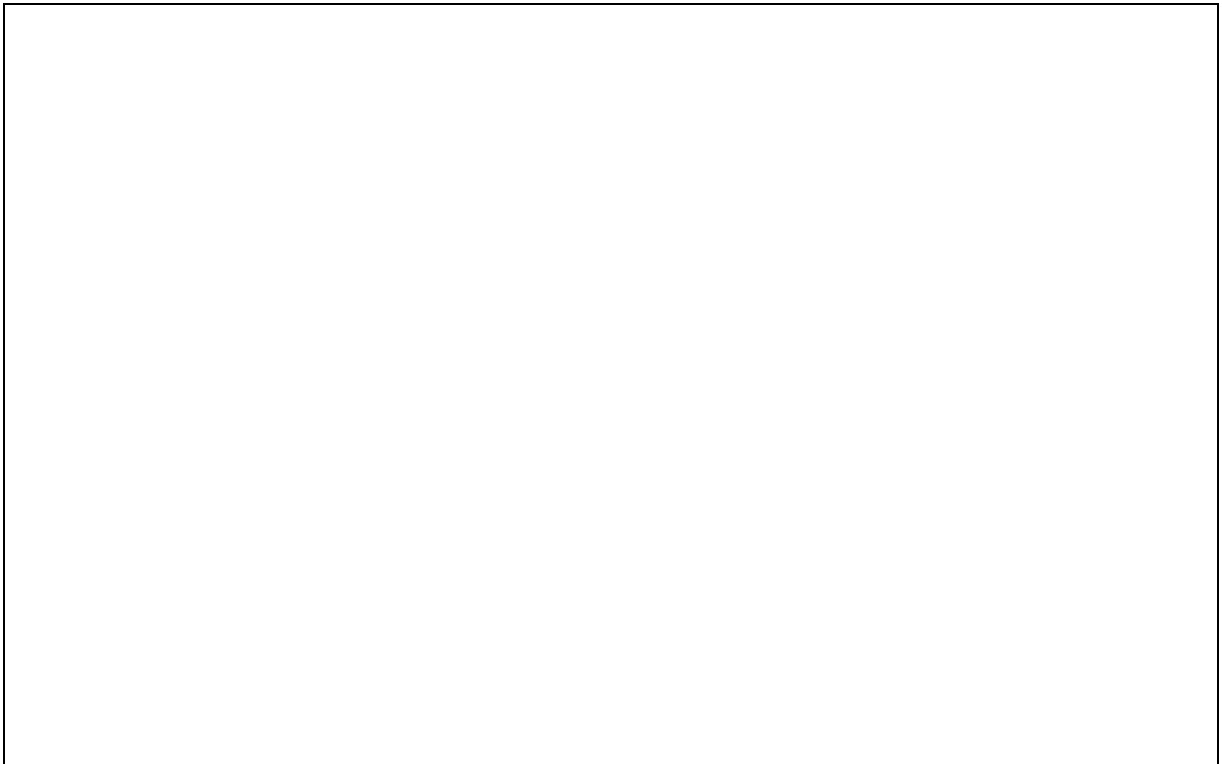


Figure 7 – The “Charmed” Telephone in Ray Dream Studio 5

4.1 The Telephone Animation Sequence

To quickly recap, the telephone animation imitates an Indian snake charmer. In this sequence, the snake is portrayed by a telephone and its enchanter is the computer, but the computer is not portrayed as a character.

4.1.1 Difficulties Encountered

As a novice animator, working with Ray Dream's complex animation and modeling techniques posed the biggest obstacle. Another problem, stemming from inexperience, was visualizing object movement and appearance in three-dimensions. Ray Dream's perspective window provides three planes of reference, a XY, a YZ, and a XZ plane. The object's outline is visible in each plane, and very confusing to the untrained eye.

To move the telephone's cord required a little scheming. Maintaining the illusion that the cord is attached to the phone base as well as the ear piece was difficult to achieve. Initially, the phones components, the cord, the base, and the receiver, were grouped together. Grouping objects molds them together as a single object. Thus the cord, the base, and the receiver would move in unison as a rigid structure. Grouping the phones components together does not generate the intended image. The base, receiver, and the cord should move independent of each other. Moving the cord required remodeling it for each keyframe. First, the ear piece was moved to the proper location. Next, the cord was manipulated using the free form modeler. Placing the end and the beginning of the cord in the correct location was accomplished purely by trail and error. Determining how the cord in the modeler related to the cord in the scene was very difficult. This relationship is

shown in Figure 8 and Figure 9. These pictures indicate the view of the cord in the scene and in the modeler. In the scene, the cord has many kinks and curves. In the modeler, the same cord appears as a hook. The relationship between the two views of the cord was finally discovered by creating extreme changes in the cord. The resulting transformation was observed and slowly the relationship became less obscure. However, the relationship is still beyond explanation. In each key frame, the telephone was viewed from all angles to ensure that the cord was still properly attached. It eventually became easier to change the cord and move the ear-piece to match, rather than the opposite. This detracted from the phone's choreography.

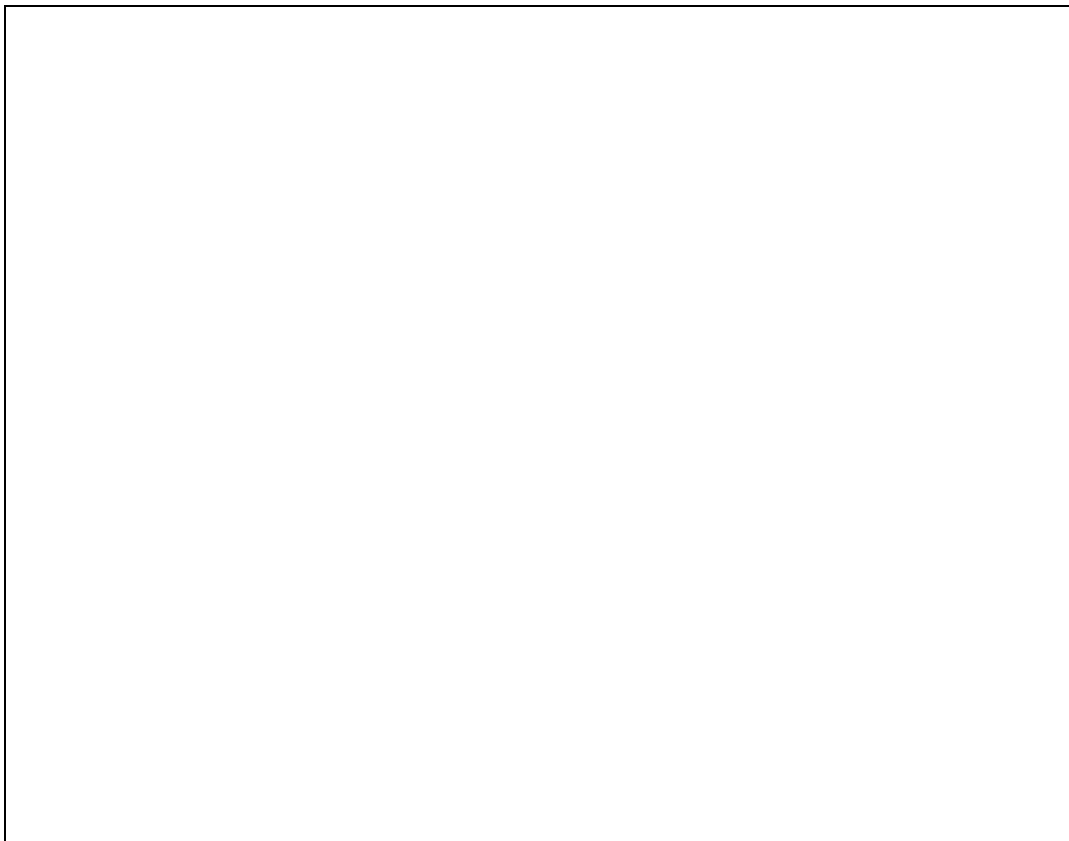


Figure 8 - View of Telephone Cord in the Scene





Figure 9 - View of Telephone Cord in the Modeler

Allowing the telephone to respond to a tune presented another problem with this animation sequence. Ray Dream does not have a button or a menu option that allows the user to play a sound file. Ray Dream does not rely upon a scripting language. So, unlike Alice, Ray Dream does not support a command that allows the user to play a sound or tune. Ray Dream animations can be wrapped in a language like Visual C++. Working with Visual C++ is a discipline within itself, and is beyond the scope of this project. For this reason, the telephone does not respond to a tune, or any other input, as originally intended.

4.2 The Software Capabilities

Ray Dream is intended for the creative individual. It offers numerous high-end modeling and animation techniques along with the option to purchase more animation and modeling features as an extensions package. Ray Dream provides artists with a platform from which to design “photorealistic” images. This animation package allows “artists to easily add the stunning impact of 3D to their

work without the long learning curve commonly associated with 3D” [Fractal Design Homepage, Online].

4.2.1 Requirements

Minimum system requirements are:

- 486 or Pentium-based computer.
- 12 MB RAM (16 MB or more recommended).
- Microsoft Windows 3.1, 95 or NT (Windows 95 recommended).
- 20 MB hard drive space for program files, plus 20 MB free disk space.
- Graphics card and monitor with minimum of 256 colors (16- or 24-bit accelerated graphics card recommended).
- CD-ROM drive.

4.2.2 The Graphical User Interface

Ray Dream Studio features a couple windows that allow the user to create and manipulate objects. These windows are the perspective, time line, browser, properties, and modeling windows. The principle window is called the perspective window, seen in Figure 10. It shows the three-dimensional workspace

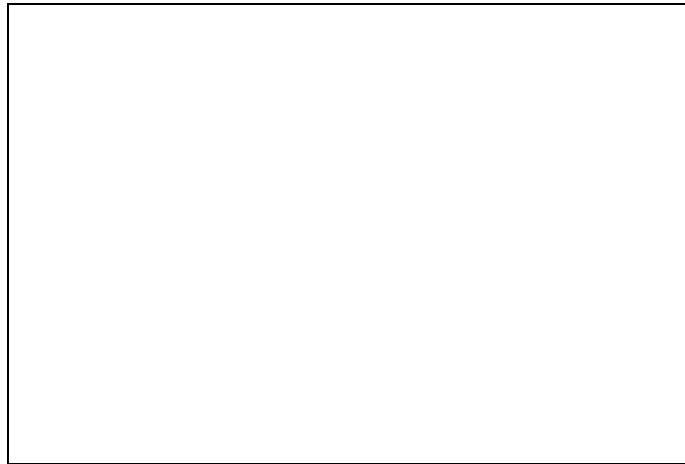


Figure 10 - The Perspective Window

where scenes are created. Objects, light sources, and cameras are placed and arranged in the perspective window. Located in the middle of the perspective window is the working box. The working box represents the XY, YZ, and XZ

planes. This gives the scene a three-dimensional quality and makes it easier for the user to visualize the scene in three dimensions. The time line window, shown in Figure 11, indicates the hierarchy in the scene and displays the occurrence of events with respect to time [Vera *et al*, 1997]. When an object is added or deleted

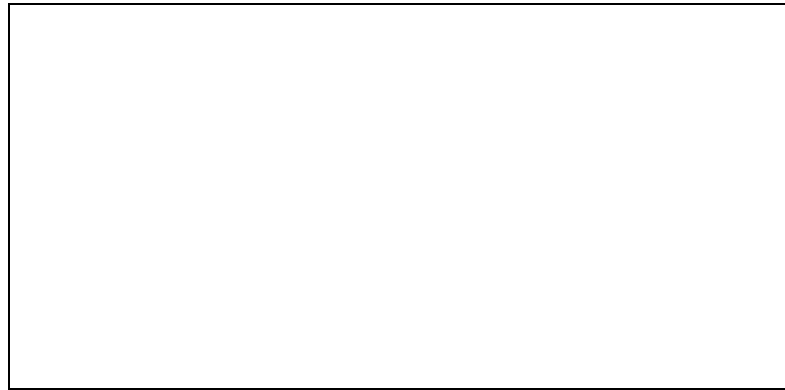
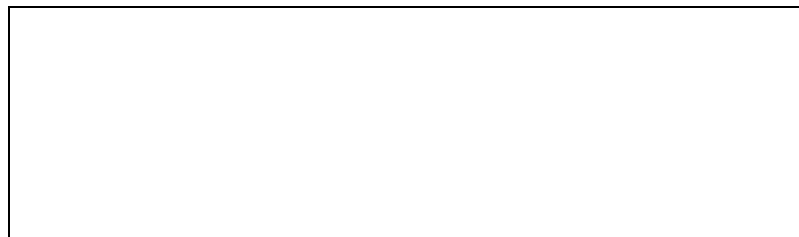


Figure 11 - The Time Line Window

from one window, the other is automatically updated. If an object is selected in the time line window, it is also selected in the perspective window and *vice versa*. This connection is convenient when trying to select small objects in complex scenes. If the object is very small, it is easier to click on its icon in the time line window than to attempt to find it in the perspective window. The browser palette, shown in Figure 12 on the following page, is a visual index of all the elements that can be added to a scene. The list includes objects, textures, lights, behaviors, and more. An item is placed in the scene by dragging and dropping it in either the time line or perspective window. The list of elements in the browser palette can be edited from a library of premodeled elements provided by Ray Dream Studio 5.



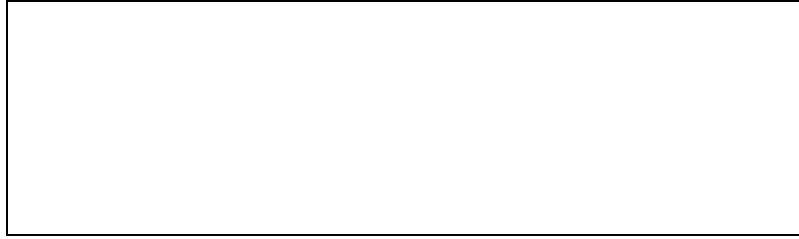


Figure 12 - The Browser Palette

The properties window, Figure 13, displays the properties of the object currently selected. These properties include orientation, position, size, and scaling. Each modeler has its own modeling window. The easiest way to access a modeling window is by double clicking on an object in the perspective window. The other option is to enter the windows from the toolbar. Each modeling window has a modeling box similar to the perspective window's working box.

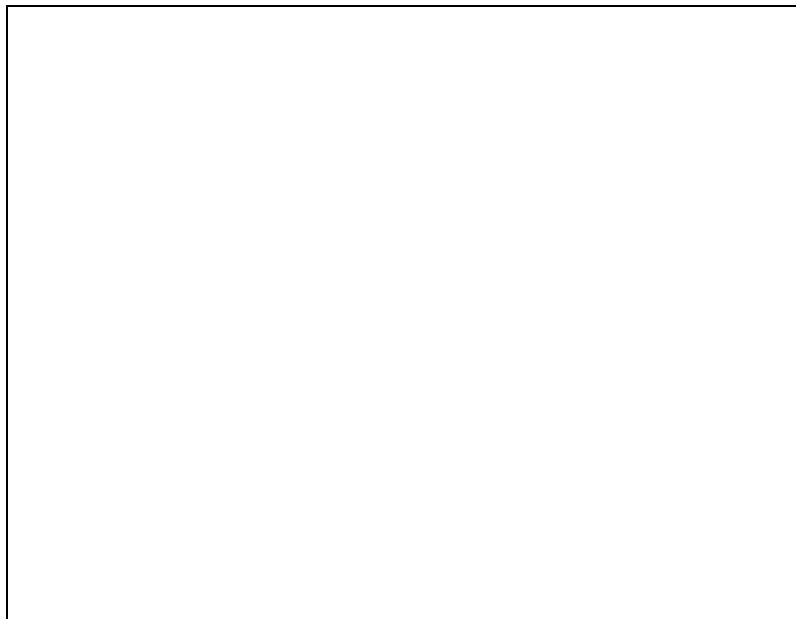


Figure 13 - The Properties Window

The modeling box consists of the three main planes. It provides the platform where the object is manipulated. In summary, the time line window is where the animator “writes the play.” The modeling windows perform the role of backstage. This is

where the costumes and props are made. Finally, the perspective window represents the stage where it all comes together.

4.2.3 Modeling Techniques

Ray Dream Studio 5 provides its user with two methods of creating objects. The first method is to build an object using primitives. Primitives are basic geometric shapes like spheres and polygons. Ray Dream also adds environmental primitives to this definition. Objects like clouds, fire, and fountains fall into this category [Vera *et al*, 1997]. The second method of object creation involves the use of a modeler. Ray Dream comes equipped with two types of modelers, the free form and mesh form modelers. Choosing between the various modeling techniques depends on the object the artist wants to create. If the object consists of geometrical shapes, primitives are used. Otherwise, a modeler is the correct plan of action. For example, look at a pen. Its basic structure consists of cylinders and cones. It is an obvious choice for construction by primitives. Now think of a jellyfish. Nothing about this creature lends itself to the use of primitives. A jellyfish is an obvious choice for a modeler.

The free form modeler allows the user to create three-dimensional objects by extruding two-dimensional objects. Extrusion is the act of adding depth to a two-dimensional object to create a three-dimensional object. The free form modeler provides numerous methods of extrusion. The simplest is straight extrusion. The two-dimensional object is swept along a straight sweep path to give it depth. Other methods of extrusion include scaling, lathing, cross sections, and complex sweep paths [Vera *et al*, 1997]. Scaling changes the size of the

object as it is extruded along a sweep path. Lathing creates symmetrical objects by rotating an object around a sweep path. Extruding with cross sections uses two different two dimensional cross sections as a framework. The modeler then extrudes between the two cross sections. Complex sweep paths entail extruding an object along a curved sweep path which gives an object the appearance of bending and curving [Vera *et al*, 1997]. The easiest way to determine which technique to use requires cutting the object in slices with an imaginary knife. The resulting slices indicate which method of extrusion to use. Identical slices indicate straight extrusion, while similar slices that vary in size lend themselves to scaling.

The mesh form modeler allows the user to create a three-dimensional object by directly manipulating the object's surface as well as using extrusion techniques. An object is generated by sweeping and extruding a primitive shape [Vera *et al*, 1997]. Then points on the object's surface are edited. Individual points are edited by clicking on the surface of the object in question. The same techniques discussed in the free form modeler can be applied to a single point. There is no legerdemain to determine what technique to apply using the mesh form modeler. The best advice is to analyze carefully each object and to proceed with thought. With a little practice, a sixth sense is developed regarding which technique to apply.

4.2.4 Animation Techniques

The artist manages the contents of the scene in the time line window. Animation is attained by moving to different points in time and making changes to the objects in the scene. These changes are called keyframe events. A keyframe is

defined as the a frame in the animation sequence where the object is at a characteristic or extreme location. The user defines a keyframe by clicking on the timeline at the point in time when the event is to take place. A marker will appear on the timeline to indicate the placement of a key event [Vera *et al*, 1997]. Ray Dream automatically fills in the transitions between the keyframes. This process is called tweening and creates not only the bulk of the frames, but the illusion of motion as well. Ray Dream lets the user tie keyframe events to real-time increments - minutes and seconds - instead of individual frames. In other words, the telephone's receiver raises from the base after three seconds instead of on the thirtieth frame. This allows the animator to focus on the appearance of his/her creation, not how many frames are in the final animation sequence.

Coupling keyframes to time allows the animator to work at a low frame rate. A lower frame rate is similar to slow motion on a video camera. This feature allows the artist preview his work and refine details that are difficult to see at a higher frame rate. When the image is rendered, the frame rate can be increased. Key events are not engraved in stone. Ray Dream allows the user to move key events to different points on the timeline, to alter them, or even to delete them. All properties can be animated in this fashion. Typically, properties define position and orientation. However, they can also refer to unconventional properties like an object's texture, the camera's zoom, or the light's brightness [Vera *et al*, 1997]. Remember that the camera and lights in a scene are considered animatable objects.

Rendering the scene is the final step in completing the animation process. Rendering is the process of adding realism to computer graphics by adding three-

dimensional qualities such as shadows and variations in color and shade. Ray Dream Studio relies upon ray tracing to accomplish this task, hence its name. Ray tracing simulates the path of a single light ray as it would be absorbed or reflected by various objects in the image. To work properly, the artist must specify parameters of the light source (intensity, color, etc.) as well as all the objects (how reflective or absorbent the materials are) [Vera *et al*, 1997]. Setting these parameters is accomplished by a pull down menu. Once the variables are set, rendering the scene is attained by the click of a button.

Ray Dream fixes the hierarchy during animation. It cannot be changed. It is not possible to add or remove objects during the course of an animation. Ray Dream also prevents the user from altering the dependencies within the hierarchy during animation. In other words, properties may change over time, but the objects themselves can not. For example, a chair can not become a couch over time, but its shape can be changed by stretching, rescaling, or deforming it in any other way possible. These restrictions are enforced because Ray Dream uses the hierarchy schema to internally manage the scene and objects during the course of an animation [Vera *et al*, 1997].

Deformers are special techniques that allow the animator to manipulate objects in a way that would normally require extensive rearranging and remodeling. For example, an explode deformer can be applied to an object. This obviously saves many laborious hours of modeling required to achieve that same effect. Some of popular deformers are bend, twist, stretch, explode, and punch. The bend, twist, and stretch deformers were used extensively to animate the

telephone cord. Deformers can also be animated [Vera *et al*, 1997].

5. Recommendations and Conclusions

Alice and Ray Dream Studio 5 are representative of the wide variety among today's graphical software. This project used a telephone animation sequence as the basis of comparison between these two software systems. This sequence incorporates numerous facets of animation and modeling. Included in this sequence are complex objects, complex object hierarchies, response to input, and complicated timing. The animation sequence highlighted the two graphics packages' strengths and weaknesses better than imagined. Accomplishing certain tasks, like playing a sound file, was simple with one package, but nearly impossible to attain in the other. This divergence was characteristic of almost all the animation functions implemented. The following sections describe solutions to the problems encountered and recommendations for future releases of the software.

5.1 Alice

Alice was designed to be "easy-to-learn authoring software for 3D graphics" [Alice, 1997]. The creators of Alice accomplished this goal. Animation techniques are constrained to simple scripts in Python. First-time users require very little effort to become familiar with keywords and start animating. Alice's ease of use its greatest strength, but also one of its drawbacks. Animating complex movements like stretching and twisting becomes very involved and complicated. Including keywords like stretch or twist would greatly increase Alice's usability.

One of the problems encountered in Alice was relying on trial and error to move the purple dinosaur between the Pavilions in the Lawn tour. The actual

coordinates of the dinosaur can be determined by the following function:

```
object.getposition()
```

Here, the purple dinosaur is the object. This gives the user the coordinates needed to use the `moveto()` function. The `getposition()` function works well, but is tedious to perform frequently. Creating a window that continually displays an object's coordinates with respect to the scene would be very useful. On the same note, adding a grid would also help prevent guessing games. A grid would enable the user to visually determine the distance between and location of objects. The grid's visibility must be toggled on and off. This feature is crucial for the appearance of the final animation. It would be unsightly to have a grid on the ground at all times during any animation sequence.

The most problematic bug encountered in Alice is its method of playing sound files. Once Alice requests a sound file, it cannot be stopped or interrupted. The easiest solution would be to force Alice to wait for a signal from the sound system to resume execution. The sound system would generate the message once the sound file has finished playing. This trick will prevent Alice from executing more commands before the previous command has completed. To halt the sound system once it has been activated is a little more complex. The sound system must be open to input while executing a sound file. Altering the sound system's behavior will allow the sound system to receive and act upon a message to stop the execution of a sound file. Currently, when a sound file is executing, the sound system locks out the rest of the world. This implementation detail must be changed.

The most serious bug encountered in the Alice software is related to its reuse of memory. It appears that Alice does not correctly reset portions of its memory upon each compilation. Initially, this does not pose a problem. However, it is not long before the performance of the software begins to suffer. In the animated Lawn Tour, Alice's memory handling deteriorated rapidly. The software would memory fault every seven to ten compilation and shut down. If the animator was not mindful to save frequently, large portions of code were lost. This quirk was originally noticed within Alice's looping construct. After trying numerous ways to work around the problem, the only immediate resolution was to avoid the use of a loop within the code. During the implementation of the telephone animation sequence, this memory quirk was noticed again. Fortunately, Alice's performance never deteriorated to the point of a shutdown, but, the cord did not perform the same upon each recompilation. It became very difficult to determine if the poor performance of the cord was related to the code or the memory management. The memory idiosyncrasy was avoided by frequently restarting the Alice software. This ensured that any problem with the appearance of the telephone cord stemmed from the code and not the memory bug.

5.2 Ray Dream Studio 5

Ray Dream Studio 5 is a high-end modeling package. It attempts to provide its user with advanced modeling techniques, but a simple user interface. The home page claims that "sophisticated 3D modeling is as simple as clicking a mouse." During the implementation of the telephone animation sequence, most of the problems encountered stemmed from inexperience. However, once Ray

Dream became more familiar territory, modeling and animating became easier. Animating in Ray Dream is often as simple as clicking a button. The problem, however, is determining which button to click.

Ray Dream Studio 5 encourages the first-time user to complete a detailed tutorial. The tutorial is very fast paced and better suited for a user with a little experience with Ray Dream. To better accommodate the needs of first-time user, Ray Dream should provide an additional, simpler tutorial. The designers of Ray Dream would benefit from performing a study using novice animators. Data from such a study would indicate the needs of first-time users. Analysis of the data would enable the designers to create a tutorial specifically for beginners' needs.

Ray Dream does a marvelous job of presenting a simple, but detailed a user interface. Most of the animating and modeling options are located on toolbars surrounding the perspective window. This setup prevents the user from switching in and out of modeling and animation applications. The layout of the user interface also enables the animator to immediately see the changes he/she made. This feature helps the user become acquainted with Ray Dream's modeling and animation techniques. It is easier to develop a feel for what a certain application can do if its effects are immediately visible.

One of Ray Dream's shortcomings is its inability to have an animation respond to input. It is not uncommon for animators to require this characteristic. Currently, the only method to accomplish such interaction is to wrap the animation in Visual C++. Ray Dream's usability would be greatly increased by adding two functions. The first is the ability to have an animation respond to a mouse-button

click. The second function is playing a sound. It is rare that animation sequences are silent, and Ray Dream should accommodate this trend.

5.3 Summary

Overall, Alice and Ray Dream Studio 5 accomplish the goals they set out to attain. Alice strives to be an easy to learn while Ray Dream endeavors to provide complex functionality in a simple environment. Both software packages could improve by borrowing elements from each other.

Animating simple motions in Alice is very straightforward, but, this only whets the user's appetite to accomplish more detailed animations. Alice does not provide for this need very efficiently. The user must accomplish complex animations using the same simple tools. The lack of complex animation tools makes high-level animation very difficult. Alice must provide for the user's growing demands.

Along the same lines, Ray Dream enables its user to create amazing animation sequences, but, getting your feet wet is not easy. Initially the animator is overwhelmed with options. Ray Dream would greatly reduce its learning curve by accounting for the novice animator. Ray Dream Studio 5 must nurture the development of the animator's needs.

6. References

Computer software

1. Alice v0.9.711 (Beta). Computer software. Three-Dimensional Programming Environment, 1995. Windows 95.
2. Ray Dream Studio 5. Computer software. Three-Dimensional Modeling Environment, 1995. Windows NT.

Interviews

1. Audia, Steve. Undergraduate Architecture Student, University of Virginia, Charlottesville, VA. Personal interview, 2 November 1997.
2. Martin, Worthy. Associate professor in the Department of Computer Science, University of Virginia, Charlottesville VA. Personal Interview, 4 November 1997.

Senior Theses

1. Taylor, Terence Glenn. "AMS PRO: A Graphical Aerial Maneuver Simulation Program for Preliminary Flight Training." Senior Thesis, University of Virginia, 1997.
2. Shochet, Joe. "Interactive 3D Painting in the ALICE System." Senior Thesis, University of Virginia, 1997.

Books

- 1 Bradford, Rex E. Real-Time Animation Toolkit in C++. New York: John Wiley & Sons, Inc, 1995.
- 2 Foley, van Dam, Feiner, and Hughes. Computer Graphics: Principles and Practice. 2nd ed. Reading, MA: Addison-Wesley Publishing Company, 1990.
- 3 MacNicol, Gregory. Desktop Computer Animation: A Guide to Low-Cost Computer Animation. Boston, MA: Focal Press, 1992.
- 4 Auzenne, Valliere Richard. The Visualization Quest: A History of Computer Animation. Cranbury, NJ: Associated University Presses, 1994.

Conference Proceedings

1. Kaprow, A. 1988. "Computer graphics and the changing methodology for artist and designers." *Proceedings Siggraph '88: 1-22*.

Journals

1. Tambe, M *et alia* (1995, Spring) "Intelligent Agents for Interactive Simulation Environments." *AAAI*, pp. 15-39.

Electronic Journals

1. "Simulation." *PCWebopedia Internet Edition*. [Online]. Available from: <http://www.pcwebopedia.com/>
2. "Script." *PCWebopedia Internet Edition*. [Online]. Available from: <http://www.pcwebopedia.com/>
3. Glinert, Susan.(1997, July) "Build a New World in 3D." *Computer Shopper* [Online]. Available from: http://sbweb2.med.iacnet.com/infot...on/375/69/7835998/16!xrn_11&bkm_16
4. Laird and Soriaz, (1997, October) "Choosing a Scripting language." *SunWorld* [Online]. Available from: <http://www.sun.com/sunworldonline/swol-10-1997/swol-10-scripting.htm>

Other Internet Sources

1. *Alice: Interactive 3D Graphics for Windows 95*. [On-line]. Available from: <http://alice.cs.cmu.edu>
2. *Alice: Frequently Asked Questions*. [Online]. Available from: <http://alice.cs.cmu.edu/faq.html>
3. *Fractal Design Homepage*. Available from: <http://www.raydream.com>
4. *PIXAR Homepage*. Available from: <http://www.Pixar.com>

User Guides

1. *Alice: The Easy-To-Learn Authoring Software For 3D Graphics in Windows 95/NT*. NP, 1997.
2. Vera and Kirsher. *Ray Dream Studio 5*. NP, 1997

Appendices

Appendix A: Alice Beta Quick Reference Card

© 1997 Carnegie Mellon University.

BASIC COMMANDS

Fred.Destroy()
 Fred.Wait(4)
 Fred.BecomeChildOf(Sue)
 Fred.Show()
 Fred.PointAt(Sue)
 Fred.Hide()
 Fred.StandUp()
 Fred.CastShadow()
 Fred.SetColor(Red)
 Fred.StopCastingShadow()
 Fred.SetVisibility(0.3)
 Fred.SetTexture('Scenery/Wood.bmp')
 Fred.SetTexture(NoTexture)
 Sue = Fred.Copy()

ARGUMENTS FOR BASIC ANIMATIONS

Fred.Move(Forward, 1.3)	Forward, Back, Left, Right, Up,
DownFred.Nudge(Forward, 1)	Forward, Back, Left, Right, Up,
DownFred.MoveTo(1,2,3)	A point in space
Fred.Place(1, OnTopOf, Ground)	OnFloorOf, OnCeilingOf, ToLeftOf, ToRightOf, InFrontOf, InBackOf, OnTopOf, OnBottomOf
Fred.Turn(Left,1)	Left, Right, Up,
DownFred.Roll(Left, 1)	Left,Right
Fred.Resize(2)	
Fred.Resize(LeftToRight,2)	LeftToRight, TopToBottom, FrontToBack
Fred.SetSize(LeftToRight,1)	LeftToRight, TopToBottom, FrontToBack
Fred.Resize(LeftToRight, 3, LikeRubber)	LeftToRight, TopToBottom, FrontToBack

OTHER OBJECT PROPERTIES

Fred.SetFillingStyle(Lines)	Solid, Lines, Points
Fred.SetLightingStyle(Lit)	Lit, UnLit

Fred.SetShadingStyle(Smooth) Smooth, Flat

KEYWORDS THAT MODIFY ANIMATIONS

AsSeenBy = Sue
 Duration = 2
 Start = Yes or No
 LifeTime = 4
 Speed = 2
 RightNow
 EachFrame
 Style = Gently, Abruptly,
 StartGently, EndGently

RESPONDING TO INPUT

Fred.RespondTo(LeftMouseDown, Fred.Turn(Left)) LeftMouseDown,
 LeftMouseDownUp,
 LeftMouseClicked,
 RightMouseDown,
 RightMouseDownUp,
 RightMouseClicked,
 MouseMove, KeyUp,
 KeyDown

CREATING & CONTROLLING ANIMATIONS

Hop = DoInOrder(
 Bunny.Move(Up,1), Hop.Stop()
 Bunny.Move(Down,1) Hop.Loop()
)
 Hop.Start()
 Hop.Loop(5)
 Hop.StopLooping()

 Dance = DoTogether(
 Bunny.Turn(Left, 1),
 Bunny.Move(Up, 1)
)

Appendix B: Alice Code for Telephone Animation Sequence

Appendix C: Alice Code for the Lawn Tour

```
walk = DoInOrder(
DoTogether(
    PurpleDinosaur.leftleg.turn(up, 1/20, speed = 1/2),
    PurpleDinosaur.tail.turn(right, 1/6, speed = 1/2)
),
DoTogether(
    PurpleDinosaur.leftleg.turn(down, 1/10, speed = 1/2),
    PurpleDinosaur.rightleg.turn(up, 1/20, speed = 1/2),
    PurpleDinosaur.tail.turn(left, 2/6)
),
DoTogether(
    PurpleDinosaur.leftleg.turn(up, 1/20, speed = 1/2),
    PurpleDinosaur.rightleg.turn(down, 1/10, speed = 1/2),
    PurpleDinosaur.tail.turn(right, 2/6)
),
DoTogether(
    PurpleDinosaur.rightleg.turn(up, 1/20, speed = 1/2),
    PurpleDinosaur.tail.turn(left, 1/6)
),
Start=No,
)

wave = DoInOrder(
    DoTogether(
        PurpleDinosaur.leftarm.roll(right, 1/3),
        PurpleDinosaur.rightarm.roll(left, 1/3)
    ),
    DoTogether(
        PurpleDinosaur.leftarm.roll(left, 1/3),
        PurpleDinosaur.rightarm.roll(right, 1/3)
    ),
    Start=No,
)

MyGUI = AControlPanel(Caption = 'Lawn Tour')
MyGUI.SetColor(Plum)
MyLabel = MyGUI.MakeLabel(Caption = 'Antrime you want to...')
MyButton = MyGUI.MakeButton(Caption = 'Kill Barney',
    Command = PurpleDinosaur.Destroy
    ("sounds/death/evilhrte.wav", duration=4)),

SecondButton = MyGUI.MakeButton(Caption = 'Color the Lawn',
    Command = Lawn3.SetColor(Purple)),

ThirdButton = MyGUI.MakeButton(Caption = 'Make the Trees Grow',
    Command = DoTogether(
```

```
tree.resize(1.5),
tree2.resize(1.5),
tree20.resize(1.5),
tree3.resize(1.5),
tree4.resize(1.5),
tree5.resize(1.5),
tree6.resize(1.5),
tree7.resize(1.5),
tree8.resize(1.5),
tree9.resize(1.5),))
```

```
closeUp = DoTogether(
    camera.move(forward,1,duration=2),
    camera.pointat(PurpleDinosaur),
    Start=No
)
```

```
talk_intro = DoTogether(
    wave.loop(2),
    PurpleDinosaur.PlaySound('Sounds/Lawn/Intro.wav'),
    PurpleDinosaur.Wait(2),
    Start=No
)
```

```
talk_pav1 = DoTogether(
    wave.loop(2),
    PurpleDinosaur.PlaySound('Sounds/Lawn/Pav1.wav'),
    PurpleDinosaur.Wait(2),
    Start=No
)
```

```
talk_pav2 = DoTogether(
    wave.loop(2),
    PurpleDinosaur.PlaySound('Sounds/Lawn/Pav2.wav'),
    PurpleDinosaur.Wait(2),
    Start=No
)
```

```
talk_pav3 = DoTogether(
    wave.loop(2),
    PurpleDinosaur.PlaySound('Sounds/Lawn/Pav3.wav'),
    PurpleDinosaur.Wait(2),
    Start=No
)
```

```
talk_pav4 = DoTogether(
    wave.loop(2),
    PurpleDinosaur.PlaySound('Sounds/Lawn/Pav4.wav'),
    PurpleDinosaur.Wait(2),
    Start=No
)
```

```
)
```

```
talk_pav5 = DoTogether(  
    wave.loop(2),  
    PurpleDinosaur.PlaySound('Sounds/Lawn/Pav5.wav'),  
    PurpleDinosaur.Wait(2),  
    Start=No  
)
```

```
talk_pav6 = DoTogether(  
    wave.loop(2),  
    PurpleDinosaur.PlaySound('Sounds/Lawn/Pav6.wav'),  
    PurpleDinosaur.Wait(2),  
    Start=No  
)
```

```
talk_pav7 = DoTogether(  
    wave.loop(2),  
    PurpleDinosaur.PlaySound('Sounds/Lawn/Pav7.wav'),  
    PurpleDinosaur.Wait(2),  
    Start=No  
)
```

```
talk_pav8 = DoTogether(  
    wave.loop(2),  
    PurpleDinosaur.PlaySound('Sounds/Lawn/Pav8.wav'),  
    PurpleDinosaur.Wait(2),  
    Start=No  
)
```

```
talk_pav9 = DoTogether(  
    wave.loop(2),  
    PurpleDinosaur.PlaySound('Sounds/Lawn/Pav9.wav'),  
    PurpleDinosaur.Wait(2),  
    Start=No  
)
```

```
talk_pav10 = DoTogether(  
    wave.loop(2),  
    PurpleDinosaur.PlaySound('Sounds/Lawn/Pav10.wav'),  
    PurpleDinosaur.Wait(2),  
    Start=No  
)
```

```
talk_rotunda = DoTogether(  
    wave.loop(2),  
    PurpleDinosaur.PlaySound('Sounds/Lawn/Rotunda.wav'),  
    PurpleDinosaur.Wait(2),  
    Start=No  
)
```

```

talk_bye = DoTogether(
    wave.loop(2),
    PurpleDinosaur.PlaySound('Sounds/Police.wav'),
    PurpleDinosaur.Wait(2),
    Start=No
)

tour = DoInOrder(
    DoTogether(
        walk.loop(1),
        PurpleDinosaur.move(forward,1,duration=4),
        closeUp.loop(1)
    ),
    DoInOrder(
        talk_intro.loop(1)
    ),
    DoInOrder(
        talk_pav8.loop(1),
        PurpleDinosaur.turn(right,1/4),
        DoTogether(
            PurpleDinosaur.moveto(-0.85, 0, -5, duration=10),
            walk.loop(3),
            camera.moveto(-0.85,1,0, duration=10)
        ),
        PurpleDinosaur.turn(left,1/4),
    ),
    DoInOrder(
        talk_pav6.loop(1),
        PurpleDinosaur.turn(right,1/4),
        DoTogether(
            PurpleDinosaur.moveto(-6.73, 0, -5, duration=10),
            walk.loop(3),
            camera.moveto(-6.73,1,0, duration=10)
        ),
        PurpleDinosaur.turn(left,1/4),
    ),
    DoInOrder(
        talk_pav4.loop(1),
        PurpleDinosaur.turn(right,1/4),
        DoTogether(
            PurpleDinosaur.moveto(-12.63, 0, -5,
duration=10),
            walk.loop(3),
            camera.moveto(-12.63,1,0, duration=10)
        ),
        PurpleDinosaur.turn(left,1/4),
    ),
    DoInOrder(
        talk_pav2.loop(1),

```

```

DoTogether(
    PurpleDinosaur.moveto(-12.59,0,0,duration=8),
    walk.loop(2),
    camera.pointat(PurpleDinosaur),
    camera.moveto(-10, 1, 0, duration=2)
),
camera.pointat(lawn3.rotunda),
DoInOrder(
    PurpleDinosaur.turn(left,1/4),
    talk_rotunda.loop(1),
    PurpleDinosaur.turn(right,1/4)
),
),
DoInOrder(
    DoTogether(
        PurpleDinosaur.moveto(-12.55,0,5,duration=8),
        Camera.moveto(-12.55,1,0),
    ),
    Camera.pointat(lawn3.pav1),
    PurpleDinosaur.turn(right,1/2)
),
DoInOrder(
    talk_pav1.loop(1),
    PurpleDinosaur.turn(right,1/4),
    DoTogether(
        PurpleDinosaur.moveto(-6.45, 0, 5, duration=10),
        walk.loop(3),
        camera.moveto(-6.45,1,0, duration=10)
    ),
    PurpleDinosaur.turn(left,1/4),
),
),
DoInOrder(
    talk_pav3.loop(1),
    PurpleDinosaur.turn(right,1/4),
    DoTogether(
        PurpleDinosaur.moveto(-0.2, 0, 5, duration=10),
        walk.loop(3),
        camera.moveto(-0.2,1,0, duration=10)
    ),
    PurpleDinosaur.turn(left,1/4),
),
),
DoInOrder(
    talk_pav5.loop(1),
    PurpleDinosaur.turn(right,1/4),
    DoTogether(
        PurpleDinosaur.moveto(6.25, 0, 5, duration=10),
        walk.loop(3),
        camera.moveto(6.25,1,0, duration=10)
    ),
    PurpleDinosaur.turn(left,1/4),

```

```
    ),
    DoInOrder(
        talk_pav7.loop(1),
        PurpleDinosaur.turn(right,1/4),
        DoTogether(
            PurpleDinosaur.moveto(12.65, 0, 5, duration=10),
            walk.loop(3),
            camera.moveto(12.65,1,0, duration=10)
        ),
        PurpleDinosaur.turn(left,1/4),
    ),
    DoInOrder(
        talk_pav9.loop(1),
        DoTogether(
            PurpleDinosaur.moveto(12.63, 0, -5, duration=15),
            walk.loop(3),
            camera.pointat(PurpleDinosaur, eachframe)
        ),
    ),
    Start=No
)

tour.loop(1)
```


UNDERGRADUATE THESIS PROJECT PROPOSAL

School of Engineering and Applied Science

University of Virginia

Three-Dimensional Modeling: Support of Scientific Visualization

Submitted by

Riesa Susan de Beer

TCC 401

November 11, 1997

On my honor as a University student, on this assignment I have neither given nor received unauthorized aid as defined by the Honor Guidelines for Papers in TCC Courses

Signed _____

Technical Advisor _____ Date _____

TCC Advisor _____ Date _____

Executive Summary

When most people think of animation, providing motion to an inanimate object is most often what comes to mind. However, animation also includes time-varying position, shape, structure, color, texture, and lighting of an object [Foley et al., 1990]. Computer animation has applications in numerous areas. The most predominant categories are: science, entertainment, research, advertising, education, and simulation. The application of computer graphics and animation in scientific fields is grouped under the heading of “scientific visualization”. The basis of these simulations are derived from scientific phenomena [Foley et al., 1990].

Computers do not animate, people do. Computers only provide a vehicle through which people animate. For this reason, understanding how a 3D modeling package acts as an interface between the animator and the computer is important. Each modeling package allows a user to specify object motion and placement in a different way. There is also no conformity among modeling packages concerning the implementation of animation techniques, such as texture mapping and object hierarchies, presented to the user. “In some cases [a modeling package] will contain an embedded animation language so that the simulation and animation processes are simultaneous” [Foley et al., 1990]. This project will address this lack of conformity. It will also propose a solution based on results obtained from designing an animation sequence in different modeling environments.

Table of Contents

<u>Section</u>	<u>Page Number</u>
1. Rationale and Objectives	3
2. Review of Relevant Literature	6
3. Statement of Project Activities	9
3.1 Activities	9
3.2 Schedule - Gantt Chart	10
3.3 Personnel	10
3.4 Resources	10
4. Expected Outcome	12
Appendices	13
A. Budget and Equipment Checklist	14
B. Bibliography	15
C. Biographical Sketch of Author	17
D. Preliminary Outline of Technical Report	18
E. Impact Statement	19

1. Rationale and Objectives

Recently George Lucas re-released the Star Wars Trilogy. To entice fans to see the familiar movies again, he included scenes that had never been seen before. The reason these scenes were originally left out was lack of appropriate technology. Today's camera tricks made it possible for Jabba to leave his palace and pay Han a visit. These "camera tricks" are more commonly known as three-dimensional modeling packages. The 3D modeling packages available today do not lack in performance, but conformity. When designing these modeling packages, designers did not have a 'recipe' to follow. Consequently, the learning curve of adapting to various modeling packages is very steep.

The question remains, what animation techniques should a 3D modeling package provide a user with.?

We are no longer bounded by what we have known always as the real world. We can create our own universe... The computer allows us to order our universe differently, to unite tools, to give us new ways to solve problems. Computers support risk. They allow us the opportunity to create ideas and develop concepts, and to experiment. They store and retrieve and give us the flexibility to try alternatives and manipulations

[Kaprow, 1988].

The vehicle through which computers "support risk" and allow us to "order our universe differently," is computer animation [Auzenne, 1994]. Computer

animation has applications in numerous areas. The most predominant categories are: science, entertainment, research, advertising, education, and simulation. Use of animation in the sciences is termed scientific visualization. Advanced computer software has been used in the simulation of military exercises, weather patterns, and even biological processes. “In theory, any phenomena that can be reduced to mathematical data and equations can be simulated on a computer”

[*PCWebopedia*]. The data and equations can be programmed and then effortlessly changed if so desired. However trivial this may sound, implementation of a simulation in an animation environment can be very complex. The current three-dimensional modeling packages that support simulation each implement the necessary equations and variables differently. Conformity among these techniques would ease the burden on animators and allow for the creation of more effective simulation environments.

From a designer’s point of view, a modeling package can be used as a tool to create an instructional module. For example, a teacher may create an instructional module used by students in his or her class, (Figure 1). This instructional module can be static or dynamic. A static module does not respond to the user whereas a dynamic module responds to input from its user. A designer focuses on how the modeling package allows a user, or animator, to create such a module. What tools and applications should be provided to the ‘instructor’? The 3D modeling package should allow for ease in specifying behavior, object motion, and articulated motion. The package should also allow for ease of coordinating multiple images on screen and texture mapping.

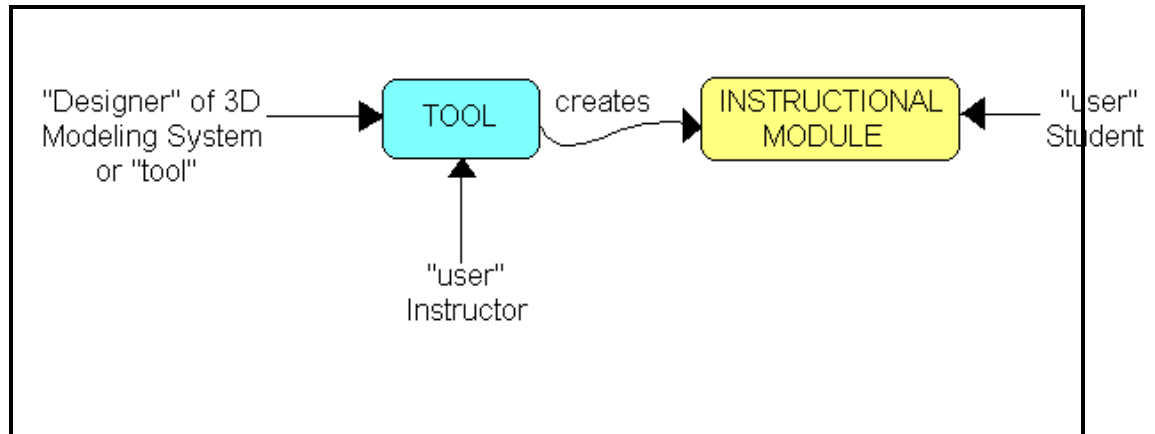


Figure 1. Use of a Modeling System[Martin, 1997]

One direction that this project will take is the design and implementation of an animation sequence in various modeling environments. This animation sequence will form the basis of comparison between the various packages. The sequence is intended to highlight the various strengths and weaknesses of the 3D modeling packages included in the study. The sequence will be based on an Indian snake charmer. However, the sequence will replace the snake with a telephone and the computer will act as the charmer.

The opening scene contains a phone sitting on a desk. A tune will begin to play in the background. As the music grows stronger, the telephone begins to respond. The phone raises its receiver and moves it like a snake's head. While the tune plays, the "charmed" telephone dances to the enchanting tune. If the music is interrupted, the receiver will drop. Otherwise, the receiver will finish its 'dance' and come to rest on the phone base. The animation sequence is an example of a dynamic instructional module because it responds to a tune. This idea was inspired by Pixar's short animated film, *Luxo Jr.*

Another aspect of this project will create a virtual tour of UVA's Lawn.

This is an example of a static instructional module. The Lawn is built in a modeling system, and the user is allowed to walk around the Lawn and inspect all the buildings. However, this tour is not dynamic. For example, the door of a building will not open as the user approaches. For the purpose of designing a tour, a static model is sufficient. The user of the virtual tour is interested in historic facts and architectural details, not visual interaction.

This project will have an impact on the animation community. A general consensus on the mechanisms provided by the software will ease the animation burden for users of 3D modeling systems. The various modeling packages currently available employ a wide variety of techniques. Familiarity with one package's conventions does not provide familiarity with any other packages. Even upgraded versions of the same modeling software have a steep learning curve [Audia, 1997]. This lack of portability is a direct result of the various design choices employed by software designers. No uniform standard for animation techniques exists. A standard method of implementation is also lacking. Providing a casual 'recipe' for modeling packages will allow software designers to learn from mistakes of those who have gone before them.

The virtual tour of the University of Virginia's Lawn will directly benefit students and prospective students. The tour will be made publicly available via the World Wide Web. From here, any person interested in gaining information on our University's rich history can access the site. The University of Virginia will also benefit from the exposure. Many prospective students are swayed to attend this establishment by the inspiring grounds. A virtual tour will allow prospective

students a chance to visit UVA without making travel arrangements.

The objectives this project will accomplish are to:

- Determine how an animation mechanism allows the user to specify behavior.
- Determine which animation mechanism to use when designing a scripting system.
- Build a virtual tour of the Lawn using the Alice software as a grand finale.

2. Review of Relevant Literature

From the earliest days, humans have used story telling for communication. Animation first appeared as a flip book in the late eighteenth century. These pictures aided the story teller, allowing him to display an event. Since then, animation capabilities have expanded to include other areas such as morphing and stereo pairs. Stereo pairs allow you to see a two dimensional object in three dimensions [Martin, 1997]. A common example is the 3D posters that have become so popular. Upon close inspection, the poster is nothing more than a few hundred brightly colored dots. However, if you stare at the poster long enough for your eyes to lose focus, an image begins to form in front of the poster. All of a sudden, a collection of dots has transformed into Lady Liberty. Non-traditional animation is a discipline within itself, and all its aspects are not discussed here. The discussion is limited to the modern adaptation of the century old flip book, key frame animation.

To create motion in animation, a series of images are successively drawn at high speeds. Each item in an animation series is called a frame. Frames are the computer equivalent of pages in a flip book. Keyframes are frames in which the object being animated is at characteristic or extreme positions [Foley et al., 1990]. For example, keyframes include drawing the first and the last location of an object on screen. After creating the keyframes, the in-between frames or ‘blank pages’ are created implicitly. The process of filling the gaps between keyframes is known as tweening [MacNicol, 1992]. In traditional hand drawn animation, this chore fell upon the shoulders of an assistant animator. Today, however, most modeling

systems perform this tedious task. The task of animating an object approaching from a distance requires only two keyframes. The animator specifies the starting and ending locations as well as the time for the object to approach the viewer [MacNicol, 1992]. Many modeling packages allow for time specification through a time line. Another common feature is path specification. This innovation allows the animator to specify the path along which the object must travel. The user notes the start and finish time and the computer interpolates the object's motion in the given time and path constraint. Keyframe animation with a computer not only allows for the timing and positioning of objects, but lighting characteristics as well. The animator can change not only the location of the light source during and animation sequence, but the type of light source as well [MacNicol, 1992]. Using the previous example, a car can now approach from a distance, while the sun rises casting different shadows. Keyframing forms the foundation in modeling programs. Advanced techniques, such as modeling add to the power of keyframe-based animation.

What is a modeling program? A three-dimensional modeling package is a software package that allows a user to generate three-dimensional images. A modeling package serves as an interface between the animator and the computer hardware. It provides the user with a window in which to create the object and its environment. This interface is then linked to the hardware via a scripting language. "Modeling is the creation of the 3-D database which serves as the 'world' to be portrayed in a synthetic computer graphics sequence" [Auzenne, 1994]. Models can be classified as either two- or three-dimensional. Even though the image

presented on the monitor is often two dimensional, the model used to create the image is three-dimensional. “Three-dimensional animation consists of three principle functions: object modeling, motion specification, and synchronization and image rendering” [Auzenne, 1994].

Object modeling builds complex models using wire-frame or solid models. Animating a complex object, such as a human being, can be simplified by dividing the object into components [Foley et al., 1990]. For example, a human can be divided into an upper body and a lower body. The upper body can be subdivided into a right and left arm. Each arm can consequently be broken down into a lower arm and upper arm, with the lower arm consisting of a hand and fingers. If an animator wanted to animate a man raising his hand, great care must be taken to synchronize the arm’s components. Having the man’s lower arm move while his hand remained stationary would not be desirable. Object hierarchies ease the difficulty of animating complex objects. Now the animator can move a man’s arm, by simply issuing the statement, ‘raise arm.’ The hierarchical dependence that exists between the components of the man’s arm would ensure that all the components would move in unison [Martin, 1997]. In other words, the man’s upper and lower arm will only move if the arm moves. The hand will move if the lower arm moves, and finally the fingers will move if the hand moves.

Creating a realistic three-dimensional object requires more than accurate structure and motion. Surface texture is an important component in creating realistic 3D models. Textures have complex coloring patterns. One can simulate these patterns by breaking the surface down into large numbers of polygons. Each

polygon will have a unique coloring scheme to create the illusion of texture. This approach is very time-consuming and error-prone. A simpler method would be to prerender a graphic texture and store it as a bitmap, a graphic file. The bitmap can then be “wrapped” onto the surface of a polygon, taking into account its three dimensional attributes. This technique is known as texture mapping [Bradford, 1995]. Mapping a rectangular bitmap onto an arbitrary polygon requires only a portion of the bitmap. The bitmap can either match the shape of the polygon or have the same number of vertices. A list of texture coordinates contains this pertinent information. A coordinate in the bitmap maps to a vertex in the polygon. The coordinate-vertex assignment is performed by the modeling system as a standard feature [Bradford, 1995].

One aspect of creating believable animation is timing. A monster racing down the hall at a terrifying pace of one pixel per hour does not have the desired effect. The frame rate is the rate at which a new frame is presented. If the frame rate is adequate, the viewer’s mind will fuse together all the images to create a single fluid image. Typically frame rates in the range of 15 to 30 frames per second are sufficient. However, if an animation series has a frame rate lower than 10 frames per second, the image will appear jumpy [Bradford, 1995]. A common trade-off in modeling systems is real-time versus accurate animation. A complex object requires more time to move, or has a lower frame rate, than a simple object. The reason behind this observation is that a simple object has less associated information to display or update in each frame. However, a monster racing down the hall without any texture is also undesirable. A compromise must be found between detail and speed.

In 1986 PIXAR, a graphics company that originated from Lucasfilm, animated a short film called *Luxo Jr.* This short film was one of the first computer animations to be nominated for an Academy Award [Auzenne, 1994]. The plot is simple, two desk lamps jump around a desktop exploring their environment. Specifically, the ‘young’ lamp plays with a ball. Luxo Jr. bends over, pushes the ball, and then watches the ball roll away. As the ball rolls away, it is illuminated by the lamp’s light. This film was one of the first sequences to illustrate the power of computer graphics. It presents itself as a case study. This short film combines all the previously mentioned techniques to give life to a small desk lamp. Even though the software on which *Luxo Jr.* was rendered predates the current software, it is still an excellent example of how to use animation mechanisms.

Within the animation community, a general consensus that the above techniques are fundamental to creating three-dimensional graphics exists. However, no set standards of how a modeling package should implement these techniques exists. How an animation mechanism allows the user to specify behavior varies among modeling systems. Which animation mechanism to use when designing a scripting system is also an area of variance. Conformity regarding these issues will not only simplify using a modeling system, but simplify designing one as well.

3. Statement of Project Activities

In order for this project to succeed, clearly defined goals and objectives are essential. Reading and researching relevant literature forms the first step. Understanding the current modeling techniques and how they have developed is important. Knowledge of the required hardware, such as a monitor, is also fundamental. Procuring knowledge of basic modeling package operations forms the foundation for the rest of the project. The next step involves the design of a simple animation sequence. All the modeling systems involved in the study will animate this sequence. The purpose of the animation sequence is to act as a control since each modeling system will have to perform the same task. Comparing how each modeling system renders the same animation sequence and discussion of the different techniques employed will form the bulk of the study. Finally, a virtual tour of the University of Virginia's Lawn will be the grand finale. The virtual tour will form a more in depth study of how the Alice system allows a user to create and specify a 3D world.

3.1 Activities

The project activities include six categories:

1. Background Research. (3 months): familiarity with the history and the current techniques of three dimensional modeling systems is essential. Research regarding this information is an important building block for the project.
2. Design of Animation Sequence. (1 month): A standard animation sequence

will be designed to analyze the strengths and weaknesses of each modeling package. This sequence will form the basis for comparisons of the packages' abilities.

3. Implementation of Sequence on Various Packages. (2 months) This step will involve the actual implementation of the above sequence.
4. Detailed Comparisons and Contrasts. (3 months) After building the animation sequence in each package, conclusions of the various techniques used can be drawn. This area is the heart of the project.
5. Modeling Virtual Lawn Objects (4 months) The grand finale of the project will be a virtual tour of the Lawn. The Alice software will be the building environment for the virtual tour. To do this all the “objects” on the Lawn, such as the Rotunda, must first be modeled and inserted into the Alice development environment.
6. Building the Virtual Lawn Tour. (4 months) Assemble of Virtual Lawn objects to form the Lawn.

3.2 Schedule

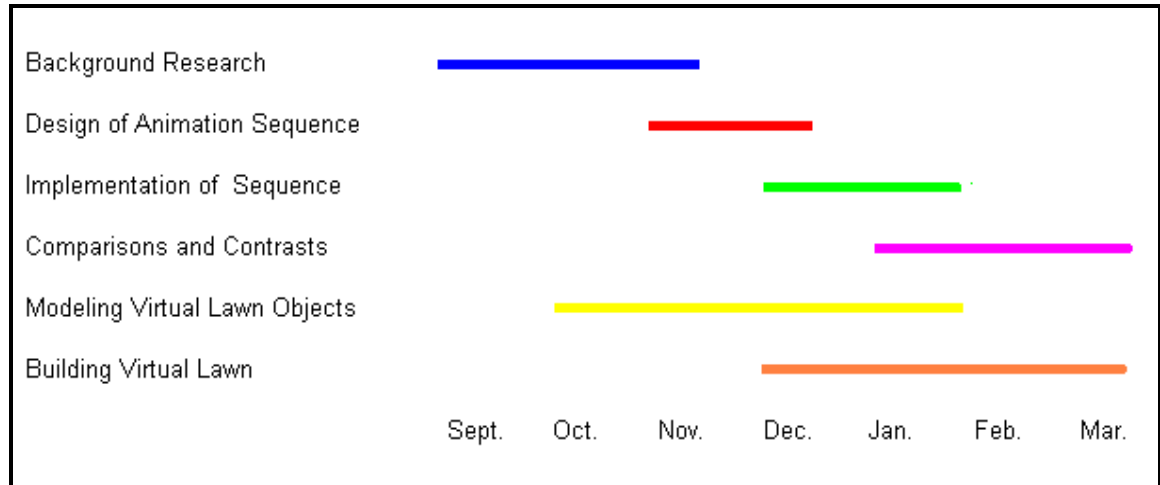


Figure 2. Gantt Chart of Project Activities

3.3 Personnel

Worthy Martin - associate professor in the University of Virginia's Department of Computer Science. He will serve as the technical advisor. His experience will be beneficial throughout the course of the project. Weekly meetings are scheduled with the advisor. These meetings will become more frequent as the project deadline draws near.

Steve Audia - a third year undergraduate student in the School of Architecture. He is well versed in building worlds on Alice. His background and experience with this system will be invaluable for this project.

3.4 Resources

The software included in the study to date are Alice, an interactive 3D graphics environment. Alice is not a modeling application. Rather it is “primarily a scripting and prototyping environment for 3D behavior”[*Alice*, online]. This, however, does not exclude the Alice software from this study. Alice is capable of reading common 3D file formats. This enables one to model an object with another modeling package and insert the object into Alice. The Alice application will still execute all the animation sequences. Alice comes with a development environment and a collection of three-dimensional objects. The main difference between Alice and a conventional modeling system, is that the Alice user has a limited selection of animation objects. A second modeling program is currently being investigated.

Alice is currently installed on a public machine in the Computer Science Department. Access to this lab has already been granted. The application only requires a computer running Windows 95 or NT 4.0 and a little imagination. Computer time is on a first-come first-serve basis, with webmasters and students using Alice having priority. Currently, efforts are underway to install the Alice software on the student’s personal computer. This will serve as a contingency plan in case the computer in the lab ever becomes unavailable.

4. Expected Outcomes

Upon completion of this project, a simple animation sequence will be rendered on various modeling systems. Animating the same sequence on each modeling package will provide a basis for comparing and contrasting the various techniques employed. This will allow a user to determine how to specify object behavior in various modeling systems. Furthermore, the study will allow a software designer to determine which animation mechanism is more effective. If, for some reason, implementing the design sequence on each modeling system is not possible, the project will lose its general application. However, all is not lost. The study will become more specific. The objectives can be applied to a modeling system as easily as they are applied to a group of modeling systems. The project will focus more in depth on the capabilities of the remaining software package/packages. Another result of the study will be the design of a virtual Lawn tour. The virtual tour will be an in depth study of animation mechanisms provided by the Alice software.

Appendices

Appendix A: Budget and Equipment List

This project requires no budget. The compulsory software and hardware have already been purchased. The software packages to be examined are installed on public machines at the University of Virginia. The Alice program is currently running on a machine in a restricted lab in the computer science department of the University of Virginia. The application requires a Windows 95 or a Windows NT 4.0 environment. Access has been granted to the restricted lab and a key obtained. Access to the machine is on a first-come first-serve basis with webmasters and Alice-users having priority. The second modeling package under consideration is currently being used by the ERICA design team. The software is installed on a public machine in the ERICA lab. Permission to use this software package has not yet been granted. If access to the modeling package is denied, another modeling package can be selected or the project can be refined to a detailed investigation of the Alice software. All research sources are available through the technical advisor and University libraries.

Appendix B: Bibliography

Computer software

1. Alice Vers. Alpha One. Computer software. Three-Dimensional Programming Environment, 1995. Windows 95.

Interviews

1. Audia, Steve. Undergraduate Architecture Student, University of Virginia, Charlottesville, VA. Personal interview, 2 November 1997.
2. Martin, Worthy. Associate professor in the Department of Computer Science, University of Virginia, Charlottesville VA. Personal Interview, 4 November 1997.

Senior Theses

1. Taylor, Terence Glenn. “AMS PRO: A Graphical Aerial Maneuver Simulation Program for Preliminary Flight Training.” Senior Thesis University of Virginia, 1997.

2. Shochet, Joe. "Interactive 3D Painting in the ALICE System." Senior Thesis
University of Virginia, 1997.

Books

- 1 . Bradford, Rex E. Real-Time Animation Toolkit in C++. New York: John Wiley & Sons, Inc, 1995
2. Foley, van Dam, Feiner, and Hughes. Computer Graphics: Principles and Practice. 2nd ed. Reading, MA: Addison-Wesley Publishing Company, 1990.
3. MacNicol, Gregory. Desktop Computer Animation: A Guide to Low-Cost Computer Animation. Boston, MA: Focal Press, 1992.
4. Auzenne, Valliere Richard. The Visualization Quest: A History of Computer Animation. Cranbury, NJ: Associated University Presses, 1994.

Conference Proceedings

1. Kaprow, A. 1988. "Computer graphics and the changing methodology for artist and designers." *Proceedings Siggraph '88: 1-22.*

Journals

1. Tambe, M et al. (1995, Spring) "Intelligent Agents for Interactive Simulation Environments." *AAAI*, pp. 15-39.

Electronic Journals

1. "Simulation." *PCWebopedia Internet Edition*. [On-line]. Available from:

<http://www.pcwebopedia.com/>

2. Glinert, Susan.(1997, July) "Build a New World in 3D." *Computer*

Shopper[On-line].

Available from:

http://sbweb2.med.iacnet.com/infot...on/375/69/7835998/16!xrn_11&bkm_16

Other Internet Sources

1. *Alice: Interactive 3D Graphics for Windows 95*. [On-line]. Available

from:

<http://alice.cs.cmu.edu>

Appendix C: Biographical Sketch of Author

Cartoons have always fascinated Riesa de Beer. At a young age she was doodling sketches of famous Disney characters on scrap pieces of paper. With time, she also developed a passion for computers. The most logical way to combine these two passions is computer animation, or more specifically, three-dimensional modeling. Many modeling environments require writing code to allow the animators to give life to their creations. Many modeling packages use C++, or a similar scripting language, as their backbone. Ms. de Beer has taken extensive classes in the programming language C++ at the University of Virginia. She also plans to participate in an advanced computer graphics course in the Spring of 1998. After graduation, Riesa de Beer hopes to pursue a career with Lexmark International Inc. as a software developer. Specifically, she wants to work in their division of computer graphics software. This project presents Ms. de Beer with an opportunity to demonstrate her enthusiasm for computer animation and her desire to do continued work in this field.

Appendix D: Preliminary Outline of Technical Report

1. Title: Three-Dimensional Modeling: Support of Scientific Visualization.

2. Major Sections:
 - *Introduction/Goals* - Provide an overview of the project's goals and how to accomplish them.

 - *Design* - Describe the animated sequence to implement on the modeling systems that will function as a basis for comparison. Also discuss the virtual tour and insights gained from its implementation.

 - *Discussion* - Detailed discussion of the various three-dimensional modeling systems, highlighting the pros and cons of each system studied.

 - *Conclusion* - Summary of knowledge obtained, goals met, improvements, and future areas of research.

Appendix E: Impact Statement

E.1 Introduction:

Continual evaluation of current methods and procedures are fundamental for progress. Such a study can result in the introduction of a new and improved method. However, the end result can also be the confirmation that a current process is most efficient. Both conclusions are valid and contribute equally to the betterment of technology. Computer animation is everywhere. If the current method of modeling can be improved, the animation community will benefit in a positive manner.

However, implementing such a change can require many resources. It demands many man hours and much money to implement a major change in a software package. This investigation will directly effect users of three-dimensional modeling programs. The results of this study will also ripple through to designers of software. Finally, this study will also influence practices in computer science.

E.2 List of Areas of Impact:

1. Animation Community

- Science
- Entertainment
- Advertising
- Simulation

- Research
 - Education
2. Software Designers and Companies
 - Design of 3D modeling programs
 - Design of software dependent on 3D modeling packages
 3. Computer Science
 - Three-dimensional modeling methods
 - Teaching animation techniques

E.3 Conclusion:

The animation community, or more specifically animators, will be most affected by this study. The conclusions drawn from this study will change the daily activities of frequent users of modeling packages. Taking a step back from the user's point of view, one can determine the influence this study will have on software designers. A more efficient method of implementing three-dimensional modeling will result in the need to upgrade current modeling packages. This will require many resources from software companies and designers alike. An upgrade requires not only substantial monetary support, but many man hours as well. Also, changing a modeling system will subsequently result in altering software for which it forms the foundation. For example, 3D computer games are currently enjoying enormous popularity. Each

game makes extensive use of three-dimensional modeling systems to bring their animation sequences to life. Hence, a change in a modeling system will necessitate a change in the dependent computer game.

The concept of three-dimensional modeling was first actualized in the field of computer science. Naturally, the repercussion of affects will be felt in this field.

The principles supporting 3D modeling will be altered. This revision will be echoed through teaching methods and textbooks.

E.4 Flow Chart of Impacts:

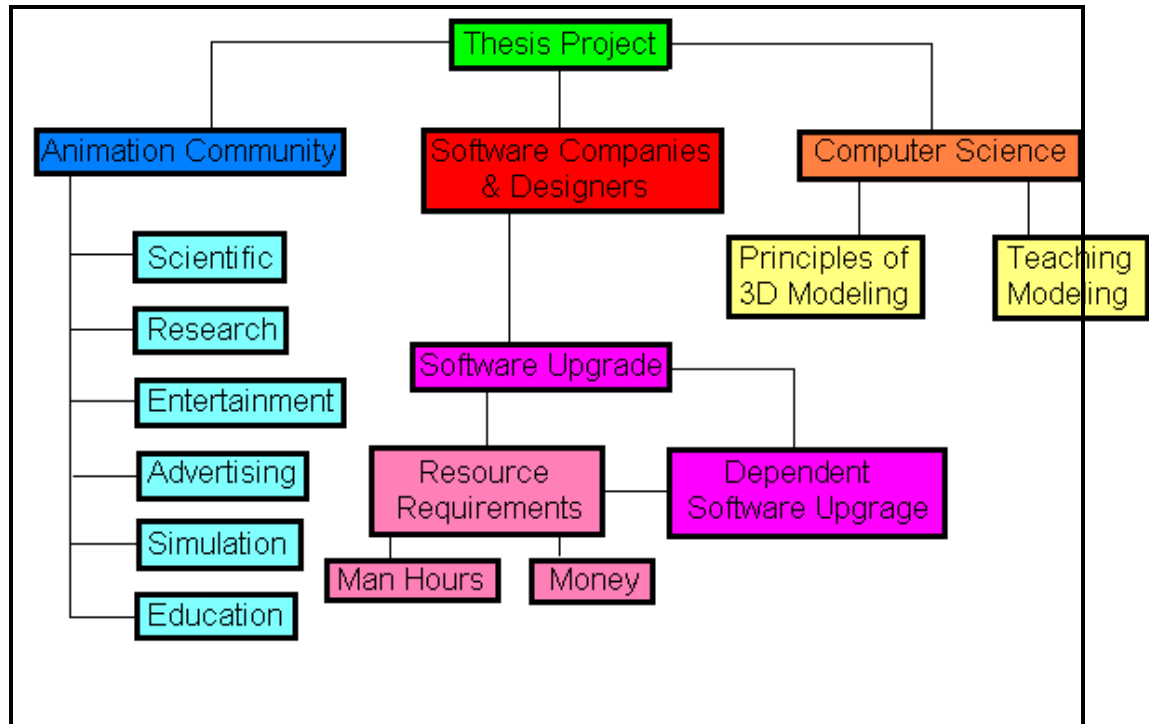


Figure 3. Flow Chart of Project Impacts