

COMMANIMATION:
A Speech-Controlled Animation System

by

Nancy Ellen Kho

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

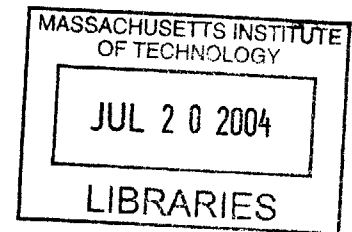
at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 20, 2004 *Elaine 2004*

Copyright 2004 Nancy Ellen Kho. All rights reserved.

The author hereby grants to M.I.T. permission to reproduce and
distribute publicly paper and electronic copies of this thesis
and to grant others the right to do so.



Author _____

Department of Electrical Engineering and Computer Science
May 20, 2004

Certified by _____

Larry Rudolph
Thesis Supervisor

Accepted by _____

Arthur C. Smith
Chairman, Department Committee on Graduate Theses

COMMANIMATION : A Speech-Controlled Animation System

by

Nancy Ellen Kho

Submitted to the
Department of Electrical Engineering and Computer Science

May 20, 2004

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

In order to make the task of animation creation easy, a different approach should be presented to the user. This thesis describes a new approach to animation creation. In this approach, the representation of animation is based on changes in state and not descriptions of the state. Users issue commands, which change the state of objects, to quickly and easily create new and interesting animations. The implemented system, called COMMANIMATION, also includes speech recognition technology, which allows for a more natural user interface. In addition to exploring animation, COMMANIMATION, a system that runs over three networked computers, provides a platform to study issues in pervasive computing, such as the use of multimodal inputs, error control, and error detection.

Thesis Supervisor: Larry Rudolph
Title: Principle Research Scientist, MIT CSAIL

Acknowledgements

First and foremost, I would like to thank my thesis advisor, Larry Rudolph, for giving me the opportunity to work on this project and for guiding me throughout the year. Through his explanations, ideas, and visions, I have learned a great deal about research and pervasive computing. Without his support, encouragement, and patience, I would not have been able to complete this project.

Thanks also to my lab mates, Albert Huang, Jessica Huang, Atish Nigam, and Debbie Wan, for their encouragement, help, and empathy this year.

I thank Emily Yan for introducing me to the technologies involved in this project and for sharing her knowledge and expertise.

I would also like to thank Hana Kim, the pioneer in creating this speech-controlled animation system.

Thanks to all the friends I have met during my time here in Boston. They have been an important part of my college years and I have been blessed by their friendships.

I give great thanks to my parents, Kim Bun and Siem Kiem Kho, who have always been there for me. I thank them for their care, support, and unconditional love the past twenty plus years, and for helping me to become the person I am today.

Table of Contents

Chapter 1 - Introduction	8
1.1 Motivation.....	8
1.2 Goals	10
1.3 Outline.....	11
Chapter 2 - Related Work.....	12
2.1 Speech-controlled systems.....	12
2.1.1 Enables easy remote access	13
2.1.2 Allows hands-free use of system	15
2.1.3 Eliminates less familiar input devices.....	16
2.1.4 Experience is more natural and real.....	19
2.2 Animation	20
2.3 Multimodal interfaces	22
2.4 Coates' speech-controlled animation system.....	24
Chapter 3 - New Approach to Animation.....	26
Chapter 4 - System Tools and Technologies.....	30
4.1 System Structure	30
4.2 Speech Builder & Galaxy	31
4.3 Flash MX 2004	34
Chapter 5 - Development of the Animation Infrastructure.....	35
5.1 Previous Work	35
5.2 COMMANIMATION.....	36
5.3 Comparison of COMMANIMATION and Flash	44
Chapter 6 - Error Control.....	46
6.1 Types of errors	46
6.2 Methods of handling transcription errors.....	47
6.2.1 Ignore	47
6.2.2 Preventing errors.....	47
6.2.3 Undo.....	48
6.2.4 Second guessing – smart handling.....	49
6.2.5 Guessing user intent.....	51
6.3 Discussion	51
Chapter 7 - The Use of Speech.....	53
Chapter 8 - Applications	56
8.1 Storybook.....	56
8.2 Diagram tool	60
Chapter 9 - Conclusion.....	63
Bibliography	66

List of Figures

Figure 1: Decision criteria for auditory or visual presentation	18
Figure 2: COMMANIMATION structure	31
Figure 3: Using relative move commands	38
Figure 4: A composite object.....	40
Figure 5: SmartChecker	50
Figure 6: StoryBook script.....	56
Figure 7: The butterfly delivers a letter from Alice to Bobby	59
Figure 8: Alice and Bob talk to each other with cell phones	59
Figure 9: Diagram Objects.....	61
Figure 10: Diagram of COMMANIMATION created with the Diagram Tool,.....	61
Figure 11: Diagram of possible cat states created with the diagram tool	62
Figure 12: Diagram with composite components	62

Chapter 1

Introduction

Intuitive and natural interfaces are the key to human-centered technology and ease-of-use. This thesis proposes a new approach to the task of animation creation and describes *COMMANIMATION*, the implementation of such an approach. In the traditional approach, animation applications present the animation as a sequence of static images, and the animator manipulates objects through the use of a keyboard and mouse. In the new approach, an animator simply issues speech commands in real-time to change objects. The proposed system allows users with little computer or animation background to quickly create interesting and meaningful animations.

1.1 Motivation

Animation is an eye-appealing and effective method for displaying information and expressing ideas. With animation, static images come to life. It has been used in a wide variety of applications, ranging from presentations to computer advertisements to web sites to games to movies.

However, creating animations is still considered to be a very specialized and tedious task. A person usually needs to go through training for software before creating animations. In the basic method of animation, a user is required to specify the objects and their attributes for each timeframe along a timeline. For example, to create the image of a ball moving from left to right, an animator might create a series of twenty frames. In each frame, the ball is placed slightly to the right of the location of the ball in the previous

frame. In the end product, the viewer sees the motion of a ball moving from the left to the right. This is quite a tedious task for just creating one motion. In a technique called keyframing, the user specifies key frames in an animation, and the computer generates the intermediate frames in between the key frames. For example, the user would create a ball on the left side in one frame, and then the ball placed on the right side in a second frame. The computer would then generate all the transition frames in between. This method is less tedious. However, animation can be even simpler!

The task of animation creation should be easy! People of all ages and without any experience should be able to create animations without any extensive training and without having to learn animation concepts such as keyframing. As many engineers know, good representations can help a person to more easily understand a situation, complete a task, and/or solve a problem. Perhaps the animation representation presented to the user in software is not the most suitable. In order to make the task of animation creation easy, we need to have an understanding of how to simplify the approach, methods, and mechanisms in software that an animator uses.

In addition to presenting a new approach in software to animation, applications can simplify the task of animation creation through the use of natural user interfaces. How do people normally communicate with each other? Some ways include gestures, facial expressions, body language, writing, and speech. Of all these, speech is probably the richest, most dominant form. Having to use computer input devices is often distracting because the user must shift his focus on the task he wishes to complete to the task of using the input devices. With speech, this is not a problem because it is so natural. In addition, language is such a rich form of expression that captures many details.

Keyboards and mice, on the other hand, are not so good for specifying details. Navigating through a menu structure or typing detailed information is somewhat tedious and awkward. Because speech is a natural form of communication for humans, using speech as an interface to computers may be much more easy and natural than using traditional input devices. Speech also opens computing capabilities to many new groups of people, such as the elderly or disabled. Speech technology has been improving and many applications using speech have been created.

For many years, people have needed to adapt to computers. Incorporating a speech interface is one way to make computers adapt to humans. In addition, through computing distributed, people are not tied to the location of specific computing devices. However, distributed computing and multimodal input introduce new sets of problems. In order to achieve human-centered computing, new research must explore these new issues of pervasive computing. In particular, error detection and control are crucial to the success of human-centered, pervasive computing.

1.2 Goals

This research work has three main goals. The first goal is to simplify the task of animation creation so that anyone with any type of animation experience can easily create new and interesting animations. COMMUNICATION is a new, more intuitive approach to animation. It also incorporates the use of speech, a rich and natural form of human expression. The second goal is to further develop the speech domain, and to examine how well a speech interface fits this application. The speech interface may also introduce new types of errors into the system, which leads to the last goal. The third and last goal is to

investigate new types of errors that may arise and the mechanisms for controlling the errors. In addition to these three goals, *COMMANIMATION* provides a platform to discover and study other issues in pervasive computing, such as error detection.

1.3 Outline

This thesis describes a new approach to animation and an implemented system called *COMMANIMATION*, which allows users to create and control animation through the use of speech commands issued in real-time. Chapter 2 describes related work in the areas of speech-controlled system, animation, and multimodal interfaces. Chapter 3 presents the new approach to animation. Chapter 4 describes the system structure, tools, and technologies used in *COMMANIMATION*.

Chapter 5 explains the changes and new commands and features available in *COMMANIMATION*. Chapter 6 discusses the errors that occur and mechanisms for controlling them. Chapter 7 focuses on the use of speech recognition and on its application to animation. Chapter 9 describes two applications of the framework provided by *COMMANIMATION*. Chapter 10 concludes this thesis.

Chapter 2

Related Work

COMMANIMATION is related to research in three main areas: speech-controlled systems, animation, and multimodal interfaces. This chapter reviews related work in these three areas. Section 2.1 describes a number of speech-controlled systems. Section 2.2 describes relevant research related to animation. Section 2.3 examines work in the area of multimodal interfaces. Section 2.4 takes a closer look at another speech-controlled animation system and examines the differences between the system and COMMANIMATION.

2.1 Speech-controlled systems

Different types of modalities are more appropriate for different types of tasks. Consequently, it is important to look at the types of speech applications that have been implemented. Research on speech recognition began about 40 years ago. Although speech technology is not perfect yet, accuracy has improved quite a bit and many applications utilizing speech recognition have been implemented. Four main advantages to using speech recognition for accessing and controlling systems include easy remote access, hands-free control, elimination of less familiar interfaces, and a more realistic experience. This section will describe a number of speech-controlled applications with a focus on each advantage.

2.1.1 Enables easy remote access

To use a system, a person often needs to have access to a computer. This is sometimes the limiting factor preventing people from accessing the wealth of knowledge and services available through the Internet. One solution is to allow systems to be accessed over the phone. Virtually everyone has a telephone these days. Combined with phone technology, speech technology allows for easy remote access to a system. This section describes systems that allow for easy access through the use of a phone.

The Spoken Language Systems (SLS) group at the MIT Computer Science and Artificial Intelligence laboratory has implemented several different speech systems, which have served as research platforms on many issues such as telephone based speech recognition, robust language understanding, language generation, dialogue modeling, and multilingual interfaces [29]. These systems allow information to be accessed over the phone. For example, JUPITER is a conversational system that provides up-to-date weather information. Another conversational system, the MERCURY system, provides information about flights available for over 500 cities worldwide and helps the user to plan and price an itinerary [29]. The system may be accessed over the web or by telephone.

Although some data can not be easily transferred over the phone, most web pages can be rendered into an audio format [4]. Accessing information on the internet through a telephone is another possibility. Several systems, such as PhoneBrowser [4] and 1-800-HYPertext [12] allow a user to browse the World Wide Web through the use of a regular telephone. These types of systems are helpful for those people who are traveling

and unable to bring along their computer, and for those who do not have any access to the internet at all.

Many companies, such as Sprint, have moved towards deploying speech recognition servers to expand their customer service coverage and control costs. Sprint has automated many customer care interactions, such as account balance, minutes remaining, billing inquiries, and call routing over the phone [10]. Customers can use the voice technology to select options rather than pressing buttons on the phone. In a case study, the Edify Corporation reports that the system handles 12 million calls per week, and 47% of the calls are managed by the automated system. The call abandon rate has been reduced from over 14% to under 0.5% with nearly zero system downtime.

AirTran is another company that has switched over to automated speech recognition phone call handling [17]. The switch took about 8 months to investigate, install, and fine-tune. Before the switch, users who called to find out about flight availability or delays spent an average of 7 minutes waiting for the call to be picked up and then about two and a half minutes for the staffer to handle the call. After the switch, the call now goes through in about 2 seconds and then a little over a minute to handle the call. This has reduced wait times, 800 toll costs, and has allowed the 650 call center employees to be transferred to sales initiatives and customer retention duties. However, a few things, such as caller interaction, do need to be improved.

A large number of workers, such as those involved in industries such as utilities, transportation, surveying, warehousing, distribution, and healthcare spend much of their time away from the office. They write on notebooks and later input this information into the computer. This process may be time-consuming and error-prone. A company called

Datria is using speech technology to replace this process [27]. With Datria's technology, a field worker may dial in with a phone to reach the speech-automated system and answer a number of questions. The data then gets directly recorded in a database. Consequently, in many cases, field service revenue and productivity have increased and costs have decreased.

2.1.2 Allows hands-free use of system

Many times in life, people are caught in a situation where they could really use an extra hand. Some examples include a parent holding a sleeping child, a cook handling multiple recipes, or a driver fumbling with a map. A speech-enabled system is one a solution to these types of situations. Speech control can provide a powerful hands-free capability to accessing a system. This section describes two applications, Ford's new SUV and Microsoft's speech engine in Windows XP.

Ford Motor Company's Model U SUV, which went on display in January of 2003, includes a conversational user interface [18]. This concept car is not in production, but is an example of what can be expected in years to come. It should enhance both convenience and safety. With the interface, drivers and passengers can control a Bluetooth wireless phone, an MP3 player, the climate control, a retractable roof, and a navigation system. Makers of the vehicle believe that speech is an easy to use and easy to learn interface. They have tried to make the interface as natural as possible, allowing the user to say things in a natural way instead of saying a specific command. As long as a person gives the system enough information, he/she does not need to say it in a specific way.

Microsoft Windows has added a speech recognition engine to Windows XP. With these new speech capabilities, a user may navigate through the operating system and some applications by using speech commands [21]. Microsoft applications, such as Word and Excel support these capabilities. However, to ensure a high accuracy of the speech recognition engine, a very good microphone must be used. The user must also train the software to adapt to his voice. Perhaps, future third-party software companies will take advantage of the Microsoft speech engine and add speech capabilities to their applications.

2.1.3 Eliminates less familiar input devices

For those people who have trouble with motor controls or vision, a speech interface would allow a user to issue commands verbally instead of having to use a mouse, keyboard, or other device to supply input. Some equipment, such as the voice activated power wheelchair designed for quadriplegics, has been in use for many years now. For those people who do not have any physical difficulties or impairments, using speech can still greatly enhance a product by making the system interface more natural and easy-to-use. In this section, two different projects, Project sentha and a speech-controlled multimedia system, will be discussed. Project sentha is intended to assist elderly people in independent living. The speech-controlled multimedia system eliminates the need for remote controls when watching television or using other entertainment systems.

An interdisciplinary team involving the Technical University Berlin, the Berlin Institute for Social Research, the German Center for Research of Aging, University of Arts Berlin, and the Brandenburg Technical University Cottbus, have been working on a project called sentha, whose goal is to create products that better serve the elderly people

who are living independently in their own homes [12]. This project has explored a wide variety of smart home technology applications, including safety and protection, comfort, communication, and environment.

In the proposed system, a normal TV and remote control are used as interfaces. However, the use of a remote control is not the most natural method and can be quite cumbersome given the number of types of commands. Some people may have trouble handling the remote control. As an alternative, the team investigated the use of a voice-controlled user interface, which would get rid of the key combinations and menu structures the user would have to remember, as well as the need to use the remote control. Some people who might be inhibited by the use of new technology may find the speech interface natural and non-intimidating.

Not all data and functions are suitable for speech. In some cases, a visual display may be more appropriate. The decision criteria on deciding the presentation type to use can be seen in Table 1 below. Speech would indeed enhance the Smart Home applications and increase user acceptance because of the natural user interface. Older people do not need to learn to use new technology and input devices.

Use auditory presentation if:

1. the message is simple,
2. the message is short,
3. the message will not be referred to later,
4. the message deals with events in time,
5. the message calls for immediate action,
6. the visual system of the person is overburdened,
7. the receiving location is too bright or dark adaptation integrity is necessary and
8. the person's job requires him to move about continually.

Use visual presentation if:

1. the message is complex,
2. the message is long,
3. the message will be referred to later,
4. the message deals with location in space,
5. the message does not call for immediate action,
6. the auditory system of the person is overburdened,
7. the receiving location is too noisy,
8. the person's job allows him to remain in one position.

Figure 1: Decision criteria for auditory or visual presentation [9]

Joseph McDermottroe, a student at the Trinity College in Dublin, has created a speech-controlled multimedia system [19]. In most modern-day homes, the family entertainment center is made up of a television, CD player, DVD player, satellite or cable system, and VCR. Having to figure out which remote control to use and then how to use the remote control may be tedious. Also, some remote controls eventually run out of batteries, which need to be replaced. Managing remote controls can be a big pain.

McDermottroe created a computer system which combines all of these separate components and replaces the remote control with the use of a voice control. Using ASP.NET, C#, an SQL database, Microsoft Speech SDK, and the Windows Media Services, McDermottroe created a system with a visual interface navigable only by voice [19].

The system was found to have few delays [19]. However, when using streaming audio and video from the web, there was a delay of a few seconds. One negative aspect that was found was that the speech recognition system used did not do well in certain situations. Some of these situations included environments with a lot of background noise. The user also needed to train the speech recognition engine in order to obtain good results. The speech recognition system also only allowed the English language to be used, which can be a limiting factor.

User evaluations were also conducted. The users found the system to be quite fast and the voice recognition worked pretty well [19]. Users also liked the flexibility to say their commands in a number of different ways. However, at some times, speech recognition did not work well, and users had to repeat their commands a number of times. Generally, users could learn to use the system quite quickly.

2.1.4 Experience is more natural and real

Speech technology may also be used to make a certain experience feel more realistic and natural to the user. Speech is a natural method of communication for people. For situations in which the reality of an experience is crucial, it makes sense to use speech. This section describes two applications, a voice activated video game and a training system for commanders in combat.

LifeLine is an adventure video game for the PlayStation 2 system released in 2004 [8]. What makes LifeLine different is that it is the only voice-activated action-adventure game. Scansoft was the company that provided the speech recognition technology. In this game, a horde of aliens have attached a space station. The player is inside the station's main control room and can see and control everything. The player

must guide a survivor named Rio through the station by giving verbal instructions using a USB headset. Forcing the user to speak aloud has a potentially immersive effect and makes the game more real.

Although this game is indeed new and exciting, the speech technology sometimes causes frustration [8]. Sometimes the speech technology is too inaccurate or too slow. In combats with other aliens, the speech recognition module interprets the spoken command incorrectly or too slowly.

CommandTalk is a spoken-language interface developed by SRI International under a DARPA sponsored project [20]. The goal is to create a training system that is as realistic as possible. CommandTalk allows simulation operators to direct and control synthetic forces in a simulated battlefield by voice using ordinary spoken English and mouse gestures, which is how commanders would control live forces. A user may create forces, control measures, assign missions to forces, modify missions during execution, and control simulation system functions. For example, the commander may issue the following commands “Charlie 4 5, speed up. Change formation to echelon right. Get in a line. Withdraw to Checkpoint 2.” CommandTalk uses the Nuance speech recognition system. CommandTalk has been installed in a number of government and contractor sites, including the Marine Corps Air Ground Combat Center.

2.2 Animation

Animation is an eye-appealing, effective method of delivering information and expressing ideas. Most of the programs for creating two-dimensional animations are designed for professionals creating intricate animations. These programs use keyframing as the dominant method of creating animations. This section describes different types of

approaches to animation. The first two, Alice and Stagecast Creator, are interactive animation systems aimed for use by children and novices. This section will also describe WordsEye, an automatic text-to-scene conversion system, and SLITHY, a script-based animation system.

Alice is a 3D graphics programming environment designed for middle school to undergraduate college students with no previous graphics or programming experience [6]. Creating an animation requires consists of two phases. First, the user creates an opening scene. Similar to `COMMANIMATION`, the user may move predefined objects from the library to the scene. In the second stage, the user creates a script that manipulates the objects placed in the scene. With its scripting language, graphical user interface, and animated three-dimensional objects, Alice can also be used to teach object-oriented programming.

Stagecast Creator is a simulation toolkit that allows children and other novice programmers to create interactive stories, games, and simulations [23]. Creator allows the users to create and modify simulations and games without a syntactic programming language. This is done through the use of programming by demonstration and visual before-after rules. Creator also provides a library of predefined objects that a user may include. Extensive user studies have shown that Creator's scene is more effective and well-received by novice programmers than traditional programming languages or scripting languages.

WordsEye is an automatic text-to-scene conversion system developed by Bob Coyne and Richard Sproat of AT&T Labs [7]. This system is similar to `COMMANIMATION` in that the user needs to learn very little in order to use the system. In

WordsEye, ordinary users can quickly create three-dimensional scenes without having to learn software, acquire artistic skills, or even use a desktop window-oriented interface. WordsEye also relies on a large library of predefined models. In this system, a user first enters a description of a scene as text. For example, a user may enter “John uses the crossbow. He rides the horse by the store. The store is under a large willow.” The system then parses, tags, and processes this text and outputs a visual scene onscreen.

SLITHY is a script-based animation system designed specifically for creating and giving animated presentations [28]. SLITHY requires the user to have knowledge of programming and of the programming language Python. To create a presentation, a user writes a program which describes the animations, rather than creating the slides directly through a graphical user interface like Microsoft PowerPoint. A presentation created in SLITHY is basically a collection of drawing objects. The three major types of drawing objects include parameterized diagrams, animation objects, and interactive objects. SLITHY is aimed at technically-inclined users and emphasizes power over ease of use.

2.3 Multimodal interfaces

People interact with other people in many ways, including the use of gestures, speech, facial expressions, and body language. Traditionally, users have interacted with a computer through the use of a mouse and keyboard. In this way, humans have had to adapt the computers. However, by augmenting a system with one or more of the more natural forms of human expression, users can interact with the computer in a more natural way. This section describes three different multimodal systems.

MATCH is a multimodal system developed AT&T Labs. MATCH allows a user to access information on New York City subway and restaurants in the area through a pen

and speech interface [13]. MATCH uses both visual and auditory output. To access information, a user may use speech commands, such as “Zoom in here” or “How do I get to this place?” The user may also use the pen to point or circle a certain area on a map. The user may even combine both of these input modalities at the same time.

The Design Rationale Group at the MIT Computer Science and Artificial Intelligence Laboratory has been working on a multimodal system which combines speech with sketching [1]. It is built on top of ASSIST, a sketching tool that lets users sketch in a natural fashion and understands a sketch as it is being drawn. The rationale behind this program is that some information specified in sketches is much more easily expressed verbally rather than through sketching. For example, it is easier to describe Newton’s Cradle with the statement “there are five identical, evenly spaced and touching pendulums” rather than drawing the system perfectly. By combining and aligning input sketching and speech, the user’s intent can be even more accurately interpreted.

The Vision Interface Group at the MIT Computer Science and Artificial Intelligence Laboratory has been working on a system which combines speech and gesture recognition in an untethered human-computer interface [15]. This system uses a stereo camera to track a user’s full-body motion in real-time. The tracking system creates and uses a pose of the 3D body model and machine learning techniques to recognize gestures. The speech and vision hypotheses are combined to guess the user’s intended action. In one application, the virtual studio, a person can use his arms and speech commands to interact with a virtual 3D world. Researchers in the Vision Interface Group have also used the gesture recognition technology to create a prototype system that allows users to guide avatars in a series of 3-D visual game worlds [25]. Results from

user studies have been encouraging. The gesture interface seems to create an immersive effect, which is a desirable quality in Virtual Reality applications [25].

2.4 Coates' speech-controlled animation system

Another project similar to our research is a speech-controlled animation system implemented built by Andrew Coates at the University of York. Coates' goal was to combine the three different technologies of speech recognition, natural language processing, and animation, and to allow an animated character to be controlled through the use of speech [5]. Coates' system contains three main modules: a speech interface, a natural language processor, and animation module.

Before implementing the system, Coates examined and compared a number of technologies [5]. IBM's ViaVoice was used for speech recognition. ViaVoice runs in two modes: Diction Mode and Command and Control Mode. The Command and Control Mode requires each application to supply its own grammar for the language of commands to accept. A user may train the application to increase the accuracy rate. The Mercury programming language was used for the natural language processing module. OpenGL was used to implement the animation. The core application was implemented in C++.

In Coates' implementation, the user uses speech commands to move a single agent in a world forwards and backwards, rotate left and right, increase or decrease speed, and to stop [5]. The set of valid paths in the world that the agent may traverse is specified in an external file before loading the program. These paths are represented as polygons or sets of polygons.

Coates also conducted a number of user tests to understand the limits of his system, to understand its feasibility as a real-world application, and compare the

differences between using the speaker-dependent and speaker-independent model [5]. Coates found that the performance, accuracy, and user enjoyment increased when the speaker dependent model was used. However, the training time for the speaker-dependent model was about 55 minutes and users found this time slow and boring. Other problems Coates points out were those dealing with delay, background noise, equipment, and user embarrassment.

In structure, Coates' system and COMMANIMATION look very similar. However, COMMANIMATION differs from Coates' system in many ways. For example, Coates uses a different set of technologies. Another difference is that Coates' system relies solely on speech as input from the user. Our system relies mainly on speech, with the mouse and keyboard as secondary input devices.

Most importantly, COMMANIMATION has a focus on creating a new approach to animation, whereas Coates' goal is to integrate all three technologies together. In Coates' implementation, the user is able to use speech commands to move a single agent in a word forwards and backwards, rotate left and right, increase or decrease speed, and to stop. COMMANIMATION defines a new approach to animation based on change and commands. A user may choose different actions and objects, and has much more control to create and change different backgrounds and control multiple characters. More powerfully, the COMMANIMATION framework allows for the creation and use of multiple types of domains and applications.

Chapter 3

New Approach to Animation

Animation can be defined as the display of a sequence of static images which appear to create a visual effect of motion. In the majority of animation programs, an animation is presented to the user as a sequence of frames, or timeslots, along a timeline. When an animator creates animations, he/she specifies the objects and their physical attributes, such as color, size, and location, for each frame along a timeline. In the finished product, the viewer sees change and motion. Because the purpose of animation is to create a visual effect of change and motion, perhaps instead of requiring the user to specify the state of objects for each moment in time, the user should specify the changes of objects over a period of time.

The idea of representing change and not state springs from Gary Borchardt's paper "Causal Reconstruction" [3]. In this paper, Borchardt describes what he calls transition space representation. The transition space representation records qualitative comparisons and changes of objects and their attributes along a time sequence of moments. Borchardt's main idea is that changes in state, and not descriptions of the states themselves, can be used powerfully to explain or make predictions about events. Borchardt implemented a program called Pathfinder, which uses the transition space representation to read simplified English descriptions of physical activity, form an internal model, and answer questions about the activity. Although animation creation is not about explaining or making predictions about events, it still may be fitting to represent animation as a series of movements or changes.

This approach is analogous to a programmer and the use of assembly language and higher level languages. In the same way that a programmer should not have to program using low level assembly language commands, an animator should not have to think of animation in terms of a series of states. Instead, the animator should think of animation in terms of higher level motions or changes.

COMMANIMATION is a new approach to animation creation. The salient idea is that animation should not be represented as a sequence of states, but as a sequence of changes, which can easily be captured by a set of commands. In COMMANIMATION, a user issues commands to create and change animation. In the old approach, a user thinks of the animation of a falling ball as a sequence of states, where each ball is located slightly lower than the ball in the previous state. However, in our approach, instead of requiring the user to think of the ball in terms of states, the user directly specifies the changes that are to occur through commands. This basic command set includes the following: create, name, place, move, delete, grow, shrink, increase/decrease animation speed, and increase/decrease movement speed. Each of these commands directly changes some attribute of an object. To create an animation of a ball falling, the user would issue the commands “Create a ball, Move ball from to y.” This is directly parallel to the animator’s goal of creating a visual effect of changes over a period of time. This new approach of issuing commands transforms the tedious task of animation creation into a much simpler, less time consuming task.

COMMANIMATION focuses on creating animations made up of predefined objects, such as an animated story, as opposed to the elaborate animation of one object, such as a flag waving the wind. In addition to supplying a set of commands,

COMMANIMATION provides the user with a basic set of objects with different types of actions. For example, a panda may eat bamboo. Animations of people, animals, and other objects are much too detailed for a user to design. With a predefined set of objects, a user may get right into creating new and interesting animations. A user may also create composite objects, which are new types of objects made up of other basic or composite objects that have already been created. This system may not be sufficient for professional programmers who desire very intricate animations. However, this method may act as a fast and powerful tool for creating sketches of animations and laying out the big picture before injecting details.

In most animation programs, a user creates a timeline which specifies when actions will take place. COMMANIMATION differs from these programs in that it allows the user to create the animation in real-time. In this case, the animation timeline is the user's real timeline.

COMMANIMATION is also unique in that it uses a speech interface. Speech is a natural method for communication for people and a speech interface eliminates the need to depend on using traditional computer input devices, such as the mouse and keyboard. Having to use computer input devices is often distracting because the user must shift his focus on the task he wishes to complete to the task of using the input devices. With speech, this is not a problem because it is so natural. The speech interface also allows extra flexibility. The user does not have to conform to some syntax, but may issue commands in many different ways. In addition, language is such a rich form of expression that captures many details. Keyboards and mice, on the other hand, are not so

good for specifying details. Navigating through a menu structure or typing detailed information is somewhat tedious and awkward for specifying parameters.

Chapter 4

System Tools and Technologies

COMMANIMATION is a system using several technologies, including SpeechBuilder, Flash MX 2004, and ActionScript. Hana Kim, a past MIT student, chose the technologies and set the structure for this system. This section provides a more detailed description of the system structure and the components of the system.

4.1 System Structure

COMMANIMATION utilizes two main pieces technology, Macromedia Flash and SpeechBuilder. Flash is used to generate the animation. SpeechBuilder is a suite of tools that allows novice developers to quickly and easily incorporate and configure a spoken dialogue system into their application. SpeechBuilder leverages the technologies in GALAXY, a conversational platform developed by the Spoken Language Systems group at MIT CSAIL. During run-time, COMMANIMATION may run on up to three computers. The Flash animation program or web browser runs on one machine, along with the frame relay, which is described in the next section. The speech server runs on a second computer. The audio capture application runs on a third computer, which may be a handheld. See Figure 2 for an image of the system design.

During run time, a user speaks into a microphone. This sound is captured by a component called GalAudio, which then forwards the data to the speech server. The speech server processes this information and passes the interpreted command and its parameters to the Flash animation server via the frame relay. Flash takes this input and

executes the corresponding action, which the user can see visually on screen. At the same time, frame relay also issues a reply, which is sent via the speech server to the output sound device. The user then hears a reply.

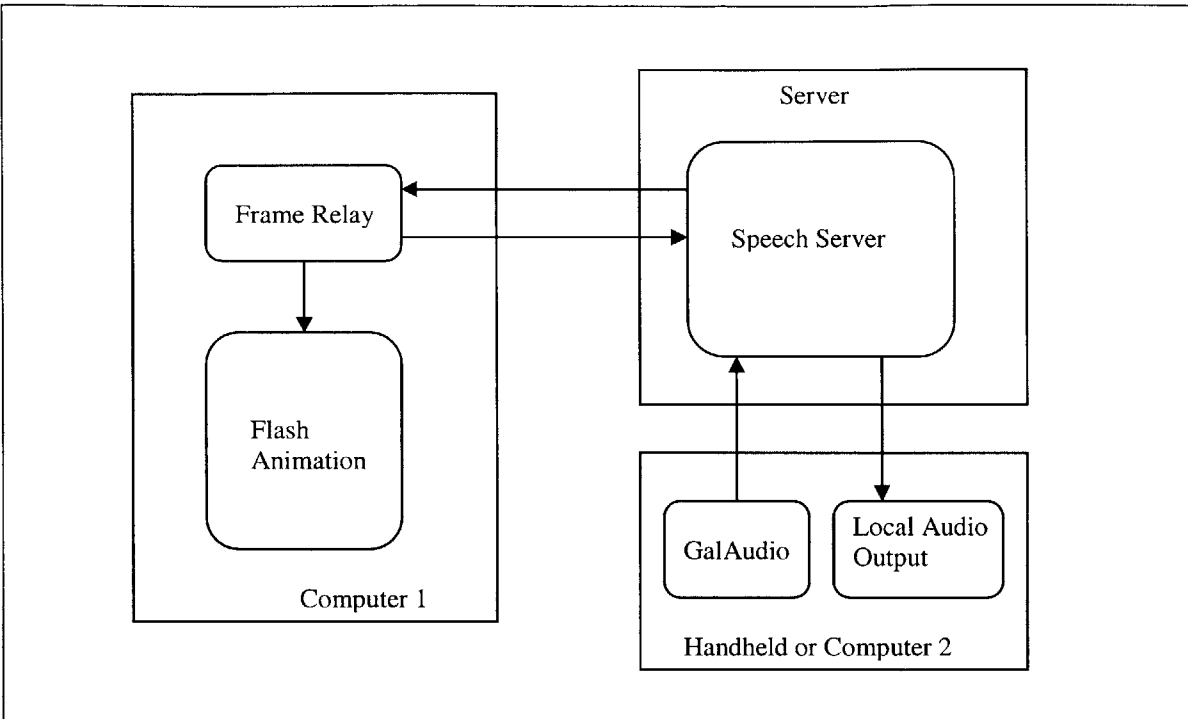


Figure 2: COMMANIMATION structure

4.2 Speech Builder & Galaxy

SpeechBuilder is a suite of tools that allows both novice and experienced developers to quickly and easily incorporate and configure a spoken dialogue system into their application [26]. It is the creation of the Spoken Language Systems Group at the MIT Computer Science and Artificial Intelligence Laboratory. Over the past decade, the SLS group has been developing Galaxy, a conversational platform. Galaxy's core technologies include as dialogue system architecture, speech recognition, language understanding,

language generation, discourse and dialogue, and speech synthesis. SpeechBuilder leverages these basic technologies. Using information specified by the developer, SpeechBuilder automatically configures the human language technology components.

Speech recognition systems are classified as being either speaker dependent or speaker independent and as either being domain dependent or domain independent. In a speaker dependent system, the user needs to train the speech engine, usually by reading a series of example sentences. In a speaker independent system, no training takes place beforehand. In a domain dependent system, the types of words and phrases that are recognized by the system are limited and predetermined. A certain number of grammars are accepted by the system. A domain independent system is not as constrained. SpeechBuilder is based on a speaker-independent, domain dependent model.

A SpeechBuilder developer first creates a speech domain by defining actions and keys and possible values for both in an XML file [26]. SpeechBuilder has a web interface that can be used for this task. Keys define a set of semantically equivalent words or phrases. See Tables 1 and 2 for examples from the SLS web site. Keys consist of the object types and object names, such as “cat,” “John,” “penguin,” or “tree.” Keys serve as non-terminal nodes in the natural language parsing grammar.

An action defines a set of functionally equivalent sentences that all perform the same operation in an application [26]. Actions are used as the top-level nodes in the natural language grammar. The example sentences for actions specify the grammars accepted by the system. SpeechBuilder generalizes the examples to accept all entries within the particular key’s class. For example, if the sentence “Draw a red circle” is specified as an example, the sentence “Draw a blue circle” will also be accepted if “red”

and “blue” belong to the same key class. In COMMANIMATION, actions consist of animation commands such as “create,” “delete,” and “move.” A complete listing of all keys and actions for COMMANIMATION is provided in the next chapter.

After the user has specified and compiled a domain, SpeechBuilder automatically configures the human language technology components. The user can then run the domain and start using his speech-enabled application.

Key	Examples
color	Red, green, blue
Day	Monday, Tuesday, Wednesday
room	Living room, dining room, kitchen
Appliance	Television, radio, VCR

Table 1: Examples of SpeechBuilder keys [26]

Action	Examples
identify	What is the forecast for Boston What will the temperature be on Tuesday I would like to know today’s weather in Denver
Set	Turn the radio on in the kitchen please Can you please turn off the dining room lights Turn on the tv in the living room
good_bye	Good bye Thank you very much good bye See you later

Table 2: Examples of SpeechBuilder actions [26]

Frame Relay is another component of the Galaxy software suite. It is a small software program that sends and receives messages across a TCP connection [16]. Frame Relay can be used to forward data from the source to a number of receivers at a destination. In COMMANIMATION, the Frame Relay runs on the same computer as Flash and passes data from the speech server to Flash.

GalAudio is a speech interface also created by the SLS group. GalAudio is a client application that allows users to connect to and interact with the speech servers. It captures sound and relays it to the speech server for processing. GalAudio may be run anywhere, such as a handheld device. During runtime, GalAudio relays the voice input to the speech server.

4.3 Flash MX 2004

Flash MX is an animation tool widely used for web games, cartoons, gaming consoles, advertisements, user interfaces, web forms, and other web applications. Because it is vector-based, Flash produces animations that are scalable without losing clarity or resolution. What makes Flash a powerful development tool is its accompanying scripting language, ActionScript 2.0, an object-oriented scripting language based on JavaScript. In our project, ActionScript is used to receive input from the speech server and to drive the animation onscreen.

One basic component to understand about Flash is the use of movie clips, self-contained movies running independently of the main timeline. In our system, a user creates new animations by using a set of predefined animated objects. Each object is represented by a separate movie clip. The movie clips have many properties and methods which can be manipulated easily using ActionScript. To create an animation, users instantiate, place, and manipulate these movie clips.

Chapter 5

Development of the Animation Infrastructure

Although `COMMANIMATION` builds upon previous work by Hana Kim, it contains a number of new animation commands and features not included before. In addition, `COMMANIMATION` further develops a new approach to animation. This project also investigates mechanisms for error control. This section will describe the changes and additions to `COMMANIMATION` from Kim's previous work, and compare `COMMANIMATION` with Macromedia Flash.

5.1 Previous Work

Hana Kim, the pioneer for creating this speech-controlled animation system, integrated all the technologies and produced a functional system for controlling animation with basic commands [14]. Kim defined a smart file naming structure. In addition, she implemented an interactive animated world in which a user can control a single penguin. In this world, the penguin can fly to one of three predefined locations including "Fatty Pizzeria," "Bookworm Library," and "Central Park." The penguin can also perform actions, such as dance, eat ice cream, and hop.

Emily Yan, an undergraduate research student, extended Kim's system such that the pre-configuration files needed by the speech software before runtime could be generated automatically. Yan also expanded the library to include new objects with new action types and added the ability to control more than one main character. In addition, Yan extended Kim's speech domain to allow placement of objects at specific coordinates.

5.2 COMMANIMATION

Although in structure COMMANIMATION looks exactly the same as the previous system, it differs in many significant ways. First, much of the code has been rewritten in the new version of ActionScript, which is much more object-oriented. Consequently, the code looks different and many new classes have been created. Second, COMMANIMATION is driven by multiple threads instead of one process. This allows for multiple actions to occur simultaneously. Third, COMMANIMATION contains many more animation commands. Users have much more control to create and control new types of animations. Fourth, users may create new objects called composite objects, which are made up of other composite objects and predefined objects. Lastly, COMMANIMATION contains a number of error control mechanisms, such as the SmartChecker and the undo feature. This section describes these changes and additions.

In the year 2004, Macromedia came out with a new version of Flash, Flash MX 2004, and ActionScript 2.0. These new versions allow many new types of actions to be implemented. ActionScript 2.0 is a full-fledged object-oriented language and supports the use of classes, interfaces, and strict data-typing. To take advantage of ActionScript 2.0, most of the previous code was rewritten.

A number of new classes were created in COMMANIMATION. These include the MovieObject, ClipArray, UndoCommand, and SmartChecker. The MovieObject class extends the MovieClip and represents the animated objects that appear on screen. A ClipArray is an array designed for holding and searching through a collection of MovieObjects. An UndoCommand represents a single, executed command. It records any

significant parameters in case the command is to be undone later. The SmartChecker is a module that checks to make sure the requested command and its parameters are a valid pair. If not, the SmartChecker offers a different recommendation. More detail on SmartChecker is provided in the next chapter.

In the system built by Hana Kim, one process handled all commands and animations. Thus, the system was limited in that only one command could be handled at a time. In addition, a user could only control one object throughout the entire animation. A new searching mechanism has been added to allow a user to control multiple objects. In addition, the system has been changed so that animation operations are handled by separate threads. This allows multiple types of animations and operations to take place simultaneously. All operations were rewritten to support use of threads.

Action(parameters)	Description	Example Sentence
Create(object type)	Creates and displays an object of the specified type	Create a cat
Delete(object)	Delete the object	Delete the cat
Place(object, x-coordinate, y-coordinate)	Place the object at the specified coordinates	Place Bob at 10, 10
Name(object, name)	Assign the object the specified name	Name the butterfly Bob
Move(object, location)	Move the object to the specified location	Move bob to the house
Perform(object, action)	Object performs special action	Bob read a book

Table 3: Old set of commands

COMMANIMATION offers many more actions and commands for creating and controlling new types of animations. The new animation commands available in COMMANIMATION are listed in Table 4. These new commands allow for greater control

of object attributes, such as the animation, movement, and size. See Tables 7 and 8 for a description of the entire speech domain.

As shown through Hana Kim's work, moving an object to an already specified location is more intuitive than moving an object to specific coordinates [14]. In addition, it also seems natural to move an object to a location relative to a previously defined location. For this reason, new move commands have been added. A user may now move an object to the right of, the left of, under, or above another object. See Figure 3 for an example created using move commands.

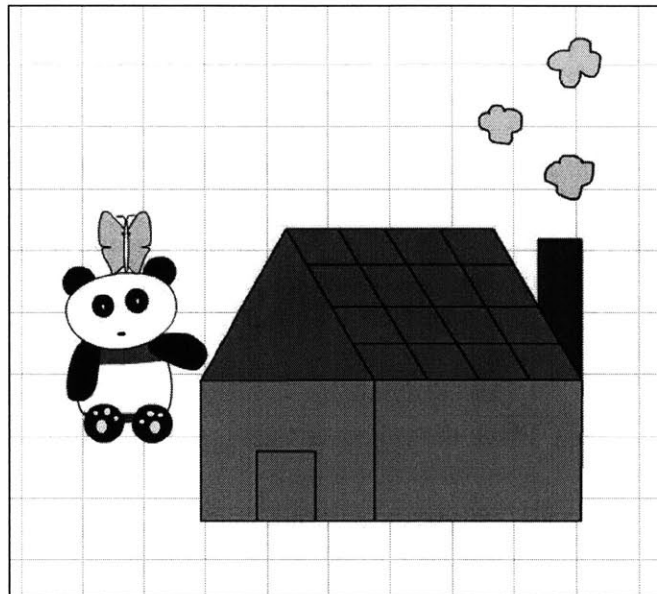


Figure 3: Using relative move commands

The panda is moved to the left of the house and the butterfly above the panda.

For the commands in which a variable, such as size or speed, is changed, the user does not specify the specific new value to change the variable. Numerical values are not as intuitive. Instead, commands which change the variable to a new value relative to the

old value are used. For example, “grow” and “shrink” are used to change the size of an object. The user may also use “grow a lot” or “shrink a lot” for a larger change.

Operations	Description
Start Animation(object)	Starts animating an object
Stop Animation(object)	Stop animating an object
Animate Faster(object)	Animate object faster
Animate Slower(object)	Animate object slower
Start Moving(object)	Starts moving an object to a location
Stop Moving(object)	Stops moving an object
Move Faster(object)	Makes object move more quickly
Move Slower(object)	Makes object move more slowly
Move to the Right of(object, location)	Move object to the right of location
Move to the Left of(object, location)	Move object to the left of location
Move above(object, location)	Move object above location
Move under(object, location)	Move object under location
Grow(object)	Increases size of an object
Shrink(object)	Decreases size of an object

Table 4: New actions in COMMANIMATION

We believe that this set of commands is close to the correct set of commands needed for a two-dimensional animation program. These commands are based on the set of attributes of an object, which includes size, location, name, animation speed, and movement speed. Other possible additional commands include those that change color or rotation.

To support creation of new types of objects, the ability to create composite objects was added. A composite object is an object composed of other objects, which can be other simple objects or other composite objects. All basic animation commands can be applied to composite objects. Figure 4 below shows two instances of a composite object of a family of cats.

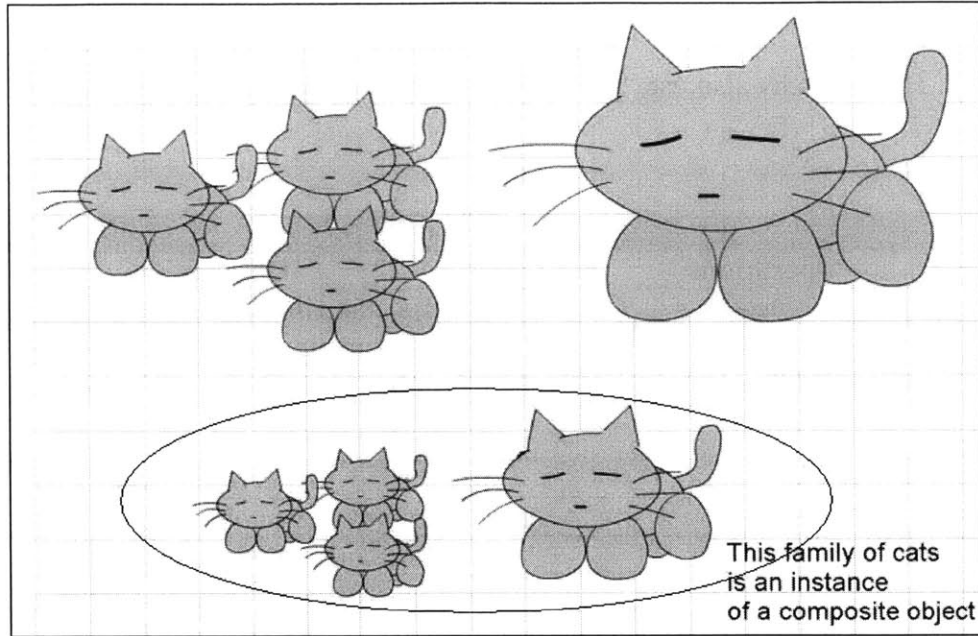


Figure 4: A composite object

To start creating a composite object, the user must first issue the “start building” command. This creates a new instance of a blank `MovieObject`. The user issues normal animation commands to create the new object. To save the current version, the command “save building” may be used so that the current object can be saved as a template for future use. After the user is done, “stop building” stops the user from building the new object and returns the user to editing the parent composite object. The root composite object is the object that contains the entire animation. Because composite may contain other composite objects, which may contain other composite objects, there is no limit to the depth of composite objects. These commands are summarized in Table 5.

Operation	Description
Start Building	Starts creating a composite object
Save Building	Saves the current composite object
Stop Building	Stops building the composite object
Create Object(identified)	Creates a new instance of a previously built composite object

Table 5: Composite object commands

The entire animation movie itself is also a composite object. When a user first starts `COMMANIMATION`, a “Start Building” command is implicitly and automatically issued. That is, `COMMANIMATION` creates a new composite object to hold the entire animation movie. All basic commands may be applied to this object.

Composite objects are helpful for creating groups or combinations of objects or alternative versions of old objects. Note that creating a totally new object with this method might be too tedious. Similarly, in real life, it is too hard and too tedious to ask another person to draw a picture exactly the way you want it. Thus, using speech commands may not be appropriate for creating new objects from scratch.

Undo commands allow the user to undo previously executed actions. As a user issues animation commands, the system records all new executed commands in a stack. To undo one operation, the user may use the “reverse one” command. To undo more than one operation, the user may use the “Start reversing” command to start undoing part commands. An intentional 5 seconds delay takes place between undoing each pair of operations. Once the user wants to stop undoing commands, he may issue the “stop reversing” command. An analogous set of “redo” commands can easily be added to the system. The undo commands are summarized in Table 6 below.

Operations	Description
Reverse one	Undo one action
Start Reversing	Start undoing previous actions
Stop Reversing	Stop undoing actions

Table 6: Undo commands

Action	Example sentences ‘[]’ represent optional words, italicized words represent keys, which may be substituted by other words
create	create (a an) <i>butterfly</i> make (a an) <i>butterfly</i>
delete	delete [the] <i>butterfly</i> get rid of [the] <i>butterfly</i>
place	place the <i>butterfly</i> at <i>fifty fifty</i>
name	name the <i>butterfly</i> <i>Larry</i>
play action	<i>butterfly go to sleep</i>
move	<i>butterfly go to [the] house</i> <i>butterfly move to [the] house</i>
move hero above	<i>butterfly move above [the] house</i> <i>butterfly go above [the] house</i>
move hero below	<i>butterfly move below [the] house</i> <i>butterfly go below [the] house</i>
move hero left	<i>butterfly move to [the] right of [the] house</i> <i>butterfly go to [the] left of [the] house</i>
Move hero right	<i>butterfly move to [the] right of [the] house</i> <i>butterfly go to [the] right of [the] house</i>
move slower	<i>butterfly move slower</i>
move faster	<i>butterfly move faster</i>
start moving	<i>butterfly start moving</i>
stop moving	<i>butterfly stop moving</i>
start animating	<i>butterfly start animating</i>
stop animating	<i>butterfly stop moving</i>
animate faster	<i>butterfly animate faster</i>
animate slower	<i>butterfly animate slower</i>
create object	create object <i>one</i>
save building	save new object save new building
stop building	stop building quit building
grow	grow <i>butterfly</i> <i>butterfly get bigger</i>
shrink	shrink [the] <i>butterfly</i> <i>butterfly get smaller</i>
start reverse	start reversing start undoing
stop reverse	stop reversing stop undoing
undo	reverse one undo

Table 7: Actions in the COMMANIMATION speech domain

Key	Key entries
formal_name	Chris Debbie Emily Jim Kevin Larry Nancy Todd Atish Jessica
object_type	butterfly cat cloud flower house panda penguin starfish telephone turtle tree
object_action	do a flip eat bamboo eat ice cream go to sleep juggle put on sunglasses read a book sing a scale start dreaming wake u
Number	0-999

Table 8: Keys in COMMANIMATION speech domain

5.3 Comparison of COMMANIMATION and Flash

COMMANIMATION's new approach to animation is much simpler and less time-consuming. This section will briefly compare COMMANIMATION with Macromedia's Flash.

Representing animation as a series of changes instead of a series of states allows the user to think of animation on a higher level. It not only makes it easier to think about animations, it also requires less work than traditional animation programs, even when keyframing is used. With language, a user can also specify a greater number of parameters in a single sentence, and the user does not need to convert the animation task into a series of mouse clicks and menu traversals.

For example, suppose a user wants to create an animation of a sleeping cat that wakes up and runs to a flower. In Flash, the user must first find and expand the "Library" listing, if not already expanded. Then, the user searches for the movie clip entry of a sleeping cat, and clicks and drags the object to the desired location on the stage. The user also searches for the flower movie clip in the Library and drags that to the screen. The user decides upon an appropriate time for the cat to sleep and specifies a keyframe at some later time in the timeline. The user must click on the appropriate time on the timeline and traverse the menu bar to insert a new keyframe. In this new keyframe, the user must delete the cat, then search for and add the movie clip of a cat running. The user then decides upon how long it should take the cat to run to the flower, chooses the appropriate keyframe at a later time, and clicks and drags the cat to the location of the house. The user then can use an action called tweening, which generates transitional frames in between two keyframes. The user uses the mouse to select on the beginning and

ending keyframe for the tweening action. Finally, he must traverse the Flash menu bar to find and apply the tweening action.

In COMMANIMATION, the user does not need to traverse any menus or remember any shortcut keys. Instead, a user issues a series of commands. To create the cat and flower animation, he may first say “Create a cat. Create a flower.” Then, he can either say “Place the cat at (100, 100)” or simply click and drag the cat to the appropriate spot. Next, he says “Cat start sleeping.” At a later time, the user says “Cat start running” to change the state of the cat. Then, the user may say “Cat move to the house.” To alter the speed of the cat, the user may say “Move faster,” or “move slower.”

In Flash, the cat and flower animation on the minimum takes approximately 10 mouse clicks, 4 mouse drags, 3 menu traversals, and 5 window searches. If the user forgets where a certain option or action is located, creating this animation takes even longer. In COMMANIMATION, this animation requires the user to issue 5 spoken commands and to click and drag the mouse twice. This is much less work. The user can say directly what he wants the objects in the animation to do, instead of thinking about the animation in terms of keyframes and animation functions. The user also does not need remember where any items are or traverse menus to find them. For a person without any animation experience, COMMANIMATION is much more intuitive and much less intimidating. However, for the experienced user who is familiar with Flash, Flash offers many more options and is much more powerful than COMMANIMATION.

Chapter 6

Error control

In any system, errors are bound to occur. Error control is an issue that must be addressed. An error should be contained and not cause greater problems, such as cause the system to crash. The system should be able to recover from the error gracefully, in a way such the user can happily continue. Errors should also be handled in a way that does not irritate the user. Adding a speech recognition capability to an application introduces a whole new set of possible error sources.

6.1 Types of errors

In *COMMANIMATION*, the types of errors encountered can be classified into three main groups. The first group consists of errors made in the hardware and software infrastructure. Examples in this group include a broken microphone, a network that is temporarily down, or bugs in any of the software systems. The second group is made of errors in transcription. These are errors made by the speech server in processing the user's spoken commands. The third group is the errors made by the user.

COMMANIMATION cannot do anything to alleviate the types of problems in the first group. Currently, the system would not be able to even alert the user of the problem. Because it runs over three different computers and uses different types of technologies, *COMMANIMATION* provides a platform to study these types of errors. Atish Nigam, another student in the Oxygen Research Group, has used this system to study error detection [22].

Transcription errors result when the system interprets the speech input incorrectly. For example, the user may say “Create a penguin,” and the system may repeatedly interpret the speech input as “Create a bunny.” Note that the system cannot distinguish between transcription errors and mistakes made by the user. A user may mean to say “Create a penguin,” but say “Create a bunny” instead. In both of these cases, the system may be able to help the situation.

6.2 Methods of handling transcription errors

Errors can be handled in a number of ways. This section summarizes a number of methods, although not all have been included in `COMMANIMATION`.

6.2.1 Ignore

The simplest way for the system to respond to an error is to ignore it. For example, if a user issues the command “Bobby eat ice cream,” and there is no Bobby onscreen, the system may just ignore the command. Although this is not the best way to handle an error, it is better than crashing. However, if there is only one object on the screen, then the user probably meant to apply the command to that one object. For these types of situations, instead of ignoring the requested operation, there is enough information for the computer to take an alternative action.

6.2.2 Preventing errors

One method of counteracting errors is to prevent errors from happening in the first place. One way to do this is to confirm all actions before actually executing them. For example, if a user says “create a penguin,” the system would ask “Are you sure you want to create

a bunny?” before executing the operation. One can imagine that performing checks like these on every single action can become annoying for the user.

Confirming a subset of the actions may be a better idea. For example, operations involving deletion may be more costly than others. Thus, confirming deletion operations may be worthwhile, while confirming other types of operations may not be. Costliness can be computed depending on the amount of work the user would be to do to undo a misinterpreted action or on the amount time the user may have to wait for the computer to undo the command. In the first option, the cost would equal the number of additional commands the user would need to issue. For example, the “shrink” command would have a cost of one because the user would only need to issue the “grow” command to undo the effects of the unwanted “shrink” command. However, a “delete” command has a much higher cost because the user would need to create, size, and place the object again.

6.2.3 Undo

To decrease the number of actions that must be confirmed, the cost of actions needs to be reduced. To make the process of undoing all actions very easy, a “reverse” or “undo” capability can be implemented. Such a capability lowers the cost of all undoable actions to one. Instead of having to create, size, and place an object again to undo the effects of the “delete” command, a user simply needs to issue the “undo” command. With the “undo” capability, the only command that needs to be confirmed in COMMANIMATION is the “exit” command.

6.2.4 Second guessing – smart handling

When a user command is not a valid command recognized by the system, the system can either ignore it or do something useful with it. SmartChecker, mentioned in the previous chapter, is an error module used to catch invalid commands and provide recommendations. The key idea is that the set of valid commands and the current state of the system may be used as knowledge to transform an invalid command into a meaningful command.

In `COMMANIMATION`, each object may have special actions associated with it. For example, a panda can eat bamboo and a cat can dream of eating fish. However, because `SpeechBuilder` overgeneralizes its language grammars, it allows the command “Panda start dreaming” to go through the system. The `SmartChecker` is a module used to check that a command and an object are a valid pair before executing the command. If an operation is not valid, `SmartChecker` supplies some recommendations. `SmartChecker` uses knowledge about the current animation environment and about the valid known objects and commands and performs simple matching. See Figure 5 for a chart describing `SmartChecker`’s algorithm.

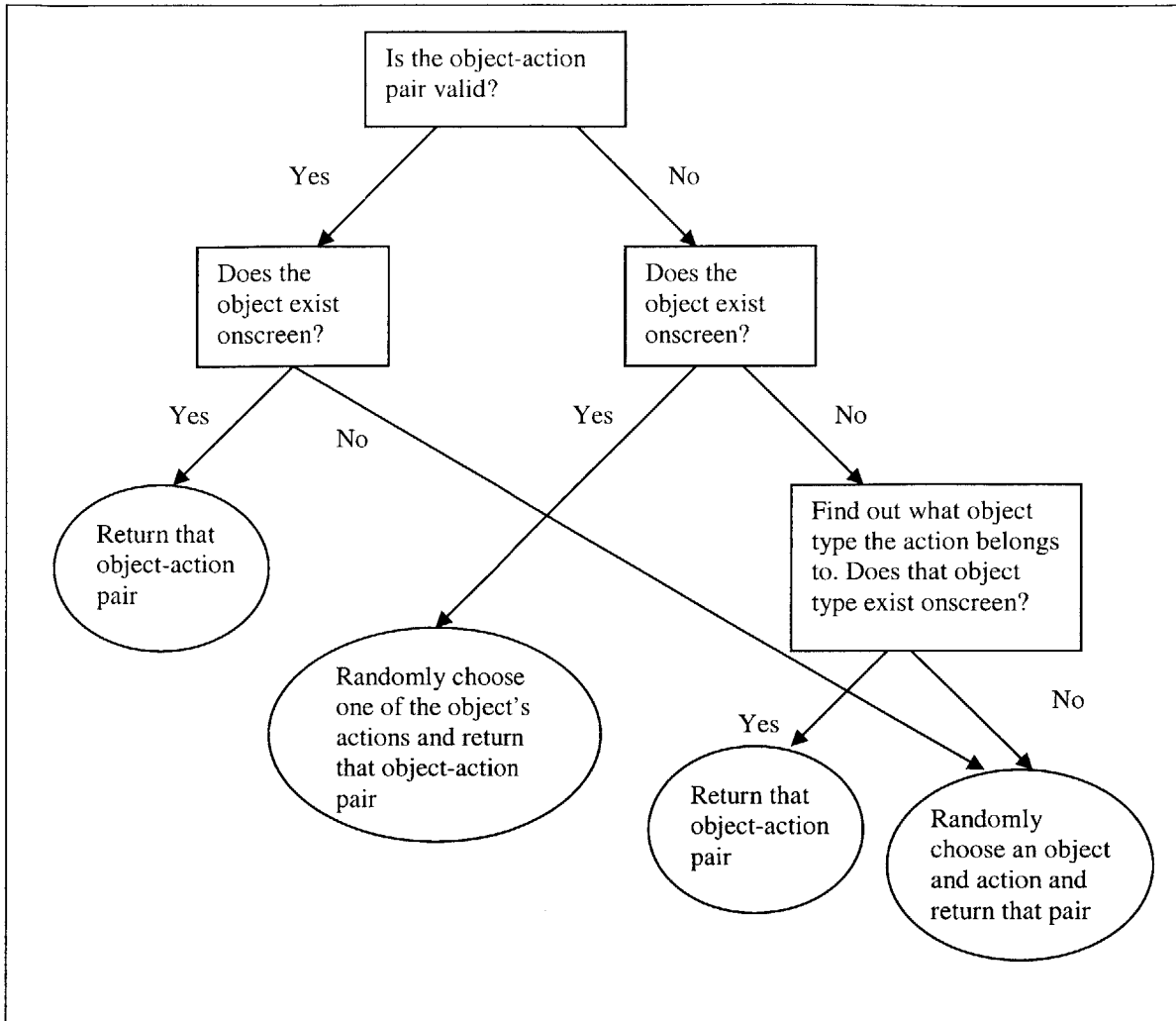


Figure 5: SmartChecker

For example, if given the command “Panda start dreaming,” which is an invalid command, SmartChecker will check to see if there is a panda on screen. If so, it may recommend the operation “panda eat bamboo.” If there is no panda, but there is a cat, SmartChecker will recommend “cat start dreaming.” If there is no cat or panda, SmartChecker will randomly choose an object and a valid action of that object to execute. Even though this action may be incorrect, it gives the user feedback and indicates that the system is still functional. In addition, some users, such as children, may find such a bug amusing.

6.2.5 Guessing user intent

A system may be able to learn from previous actions and guess the user intent. Although this method was not used in COMMANIMATION, the system can be extended to include this technique. A history of past actions can be recorded by the system. Once a user attempts to undo a command and/or repeats some command a number of times, the module for guessing user intent may kick in. For example, if a user says “make the dog sit,” and is misinterpreted as “make the doll sit” a number of times before correctly being interpreted, the system may remember this in the future. Thus, if the user undoes the command “make the dog sit,” the system may immediately guess that the user’s intent is “make the doll sit.” AI techniques, such as Bayesian nets or neural nets, could be used for calculating the most probabilistic choice.

In addition, depending on the amount of knowledge and common sense the system has, the system may also perform smart reasoning. For example, if the user issued the command “cat start barking,” the system would execute the command “cat start purring” instead.

6.3 Discussion

Of all these methods, reflexes, smart handling, and undo commands are used. By having the ability to undo commands, the cost of executing all possible commands is decreased, and confirming a user’s intent is not necessary.

A module that guesses the user’s intent, as described in section 6.2.5 was not implemented. When using this method, it is important to remember that it is not good to adapt to user errors and to build upon user mistakes. If so, a user could end up saying

“make a rabbit” in order to delete a cat. Also, COMMANIMATION is designed to be user independent. Adapting to each user’s mistakes would defeat this goal.

Another important point is on the use of abstraction of designing systems. Abstraction is an important principle in designing systems. However, it is important to not hide too many details. In this system, the speech server outputs the best choice or most likely sentence spoken by the user. If the speech server could offer more than the best choice, such as the second and third best choice, COMMANIMATION could be smarter and use that information to perform better matching.

Chapter 7

The Use of Speech

This section will describe areas in which speech recognition technology does not work so well and discuss the application of speech to animation.

Although speech technology has greatly improved after the past few years, it is still not robust enough to use for critical applications. Improved hardware and speech recognition could alleviate many problems. However, even with a very high quality microphone, if it was too far away, the accuracy decreased. When the command has many parts and is more complex, the system has greater probability of getting one part wrong, and consequently interpreting the entire command incorrectly. In choosing the command and object names, words that sounded differently were used. Words with similar sounds are often confused. For example, in our experiments, the name “Bob” was interpreted as “Todd” greater than 50% of the time. These areas of difficulty may be improved and possibly eliminated with better speech recognition software and/or hardware.

Some characteristics about speech recognition may never be improved. For example, one might avoid using a system because it disturbs the silence. Speaking commands also sometimes takes longer than just clicking and dragging on the computer. When information is very detailed, it may be tedious for the user to speak all of the commands. However, because the system is designed to allow use of the mouse and keyboard, this is not so much an issue for our system.

For some situations, speech may not be the best method. For example, when there are many similar objects, it is hard to specify which object the user wants. Using a pointing interface may work better. This could be solved by naming, but if there are many objects, the user will not want to name every object. Also, for specifying new locations, pointing may also again be more natural than using coordinates.

Although speech is indeed a more natural interface than traditional computer input devices, it must be used appropriately. If not, speech may add frustration to the user. First, before applying speech to some task, the developer should think about whether or not speech is appropriate for the task. Second, the application must use natural, intuitive commands. Third, the combination of different types of modalities can be much more powerful.

As shown through the use of COMMUNICATION, speech works well with the new approach of creating animations. First, the new approach of animation allows the user to think animation in terms of a sequence of changes instead of a sequence of states. This allows the user to think of animation on a higher level, and hides many of the small details. The user simply issues commands to create and control an animation. Because the animation is controlled in real-time, the user does not need to specify the timing beforehand. This approach also relies on a large library of predefined objects. Because this new approach is much less tedious, speech works well. Second, animation takes place according to a timeline. As mentioned before, the use of speech naturally assigns an ordering to the spoken commands. This fits animation.

As mentioned before, speech does not work as well in some areas. For example, speech is not good for specifying an exact location. Perhaps by adding a touch-screen, COMMANIMATION can become even more natural and easy to use.

The magic in COMMANIMATION is not in the speech capability, but in the commands. Language is a much richer form of expression that captures many details. The commands, which are expressed as phrases or sentences, are simple, short, and make sense to the user. Because of the nature of these commands, speech works well with this system. Speech, in turn, also makes the system feel more natural. The user does not have to think about how to transform the task into a series of mouse clicks and menu traversals, but can directly express his goal. The user can specify a command in a number of different ways. Speech is a good way to interact with animations.

Chapter 8

Applications

In addition to creating animations, the framework provided by COMMANIMATION can be used in many new and different ways. Some applications include presentations, animated storybooks, games, and simulations. As described earlier, Hana Kim implemented an interactive animated world in which a user can control a single penguin. This section will describe two new applications and the additional work required to support them.

8.1 Storybook

The first application is an animated storybook. As a user reads through the story, COMMANIMATION displays the corresponding animation onscreen if the line is read correctly. See Figure 6 below for the script and animation clips in Figures 6 and 7.

Once upon a time, there was a panda. The panda was named Alice. There was a cat. The cat was named Bobby. Alice and Bobby were friends. Alice liked to do a lot of things. Alice liked to read and Alice liked to juggle. Bobby was a lazy cat that liked to sleep all the time. Bobby wake up!

One day bobby moved far, far away. Bobby moved to the left of the tree. There was a penguin. The penguin was named Todd. Todd was Bobby's crazy friend. Todd liked to eat ice cream.

One day Alice was bored. So Alice decided to read a book. After reading this wonderful story, Alice wanted to tell Bobby about the book, but he was far, far away. So Alice decided to write a letter.

There was a Butterfly. The butterfly was named Chris. Chris worked for the US Postal Service. Alice sent the letter. And Chris made the delivery to Bobby. It took many days, but finally Bobby read the letter.

Later on, Alice got a cell phone and Bobby got a cell phone. So we can say farewell to Chris.

Alice could call Bobby whenever she wanted to talk to him. And they could clearly communicate with their cell phones whenever they wanted, even though they lived far far away. Years later, Alice fell in love with Bobby and they lived happily ever after.

Figure 6: StoryBook script

To support a new `COMMANIMATION` application, a developer may make four main types of changes. First, the developer may add new example sentences to the speech domain to allow a user to specify a command in a new way. Second, the developer may create new types of animation objects, or movie clips, in Flash. Third, the developer may add new actions and attributes to the speech domain. Fourth, new actions may require extensive coding of new ActionScript functions.

In the storybook application, only the first three types of changes were made. First, new example sentences were added to the speech domain so that a user could read complete normal sentences to produce animations instead of issuing commands. For example, instead of saying “Create a cat,” the user can say “And there was a cat.” New sentences were also added to support both the present and past tense. The new example sentences are listed in Table 9. Second, nine new Flash movie clips were created. These clips can be seen in Table 10. Third, four new character actions were added. These actions include “juggle,” “write a letter,” “read a letter,” and “got a cell phone.” Two new SpeechBuilder actions, “loves” and “calls,” were also added. These two new actions could be handled by calling one or more of the primitive animation commands. When either of these actions is executed, `COMMANIMATION` plays the appropriate clips. For example, if “Panda calls cat” is read, `COMMANIMATION` plays the movie clips of the panda and cat speaking on the phone. No new ActionScript functions were needed.

Action	Example Sentences
	Italicized words are keys
Create	There was a <i>panda</i>
Name	The <i>panda</i> was named <i>Alice</i>
Play Action	<i>Alice</i> decided to <i>read a book</i>
Move	<i>Chris</i> made the delivery to <i>Bobby</i>
Delete	Farewell to <i>Chris</i>
Calls	<i>Alice</i> calls <i>Bobby</i>
Loves	<i>Alice</i> loves <i>Bobby</i> <i>Alice</i> fell in love with <i>Bobby</i>

Table 9: Additions to speech Domain for the StoryBook application

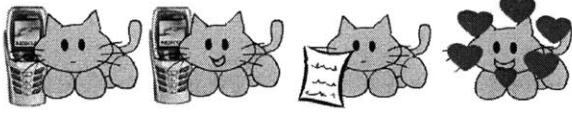
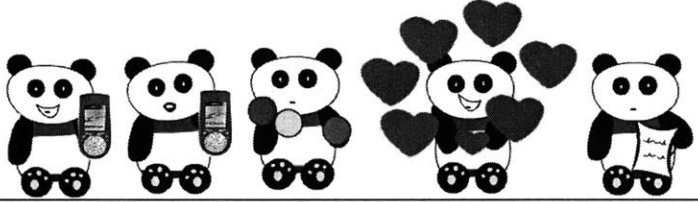
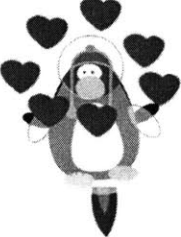
Description	Image clip
cat got a cell phone, cat talks on phone, cat reads letter cat is in love	
panda got a cell phone, panda talks on cell phone, panda juggle, panda is in love panda writes a letter	
penguin is in love	

Table 10: New movie clips for StoryBook application



Figure 7: The butterfly delivers a letter from Alice to Bobby

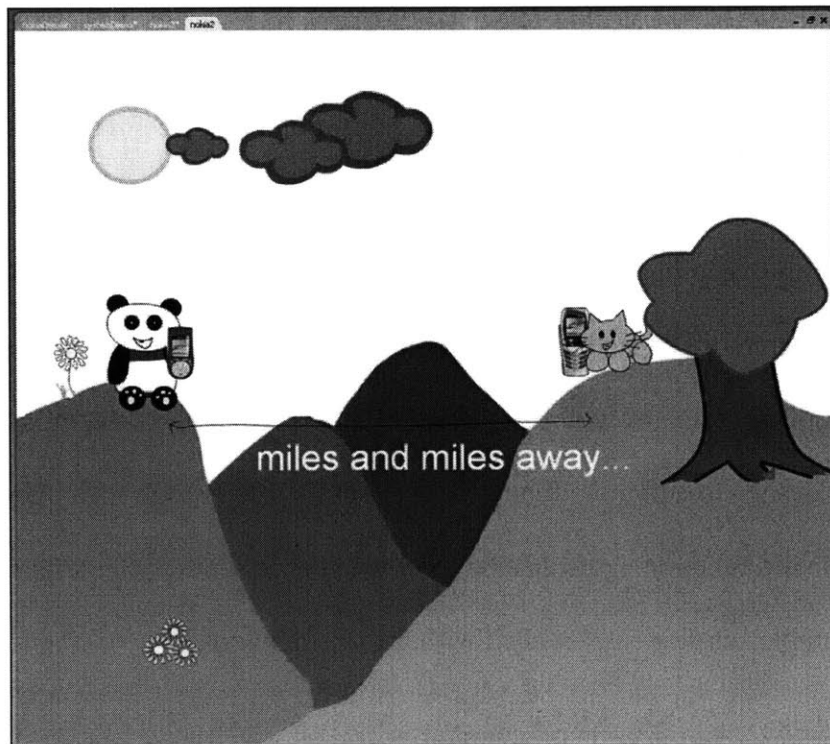


Figure 8: Alice and Bob talk to each other with cell phones

8.2 Diagram tool

The second application is a tool for creating diagrams. With this tool, a user can create and connect a number of components to make a simple diagram.

To create this tool, seven new movie clips and three new actions were added. The clips, which include three component shapes and four arrows, are listed in Figure 9. If rotation was included as a basic attribute and commands for changing rotation were added to the basic animation command set, only one clip of arrow would be needed. The first command added is the “connect” command, which allows a user to create a unidirectional link between two components or objects. Second, the “create a composite component” allows the user to create a component that can contains other subcomponents. This is similar to the command used to build a composite object. However, instead of using an invisible container to hold the objects, a component container holds the objects. The third command is the “change shape” command.

Most of the additional work for creating this tool took place in the ActionScript code. Because of the nature of diagrams, new functions for the three new commands were created. The primitive animation functions were not enough. Unlike animation, objects in diagrams are much more dependent on each other. For example, the length, beginning point, and end point of an arrow directly depends on the two objects that it is connecting. If a developer desires to add more functionality to this diagram tool, many new functions, as well as some old functions, may need to be written or re-written. See Figures 10, 11, and 12 for examples of diagrams created with the diagram tool.

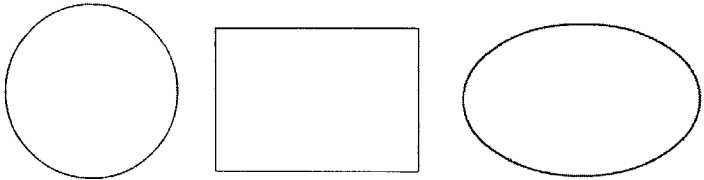
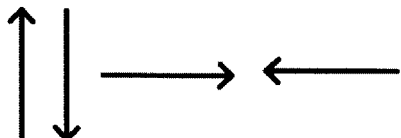
Description	Image Clip
Component	
Arrow	

Figure 9: Diagram Objects

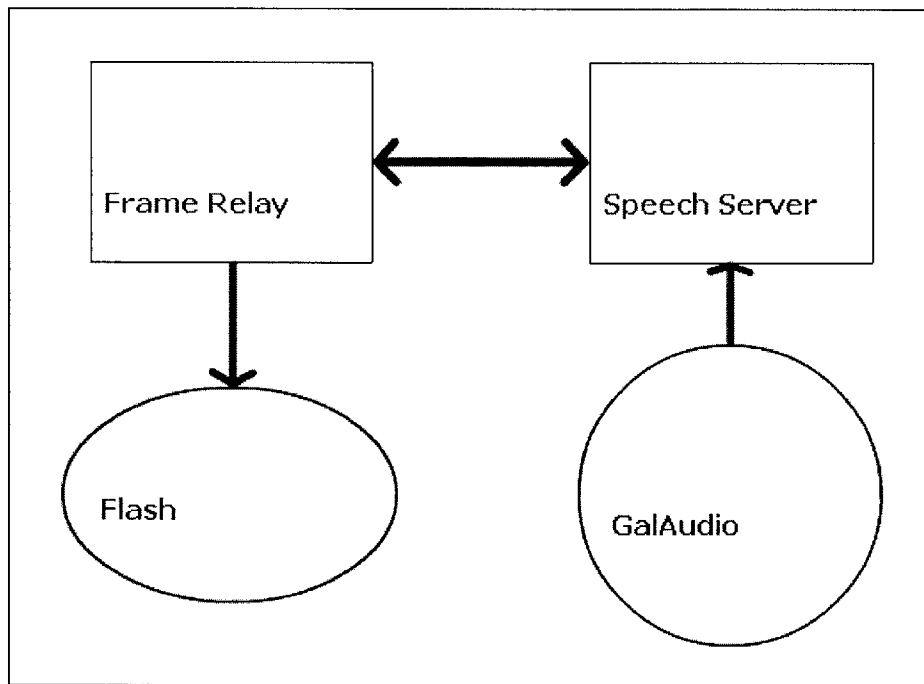


Figure 10: Diagram of COMMANIMATION created with the Diagram Tool, created by a series of Create, Change Shape, Place, Name, and Connect commands.

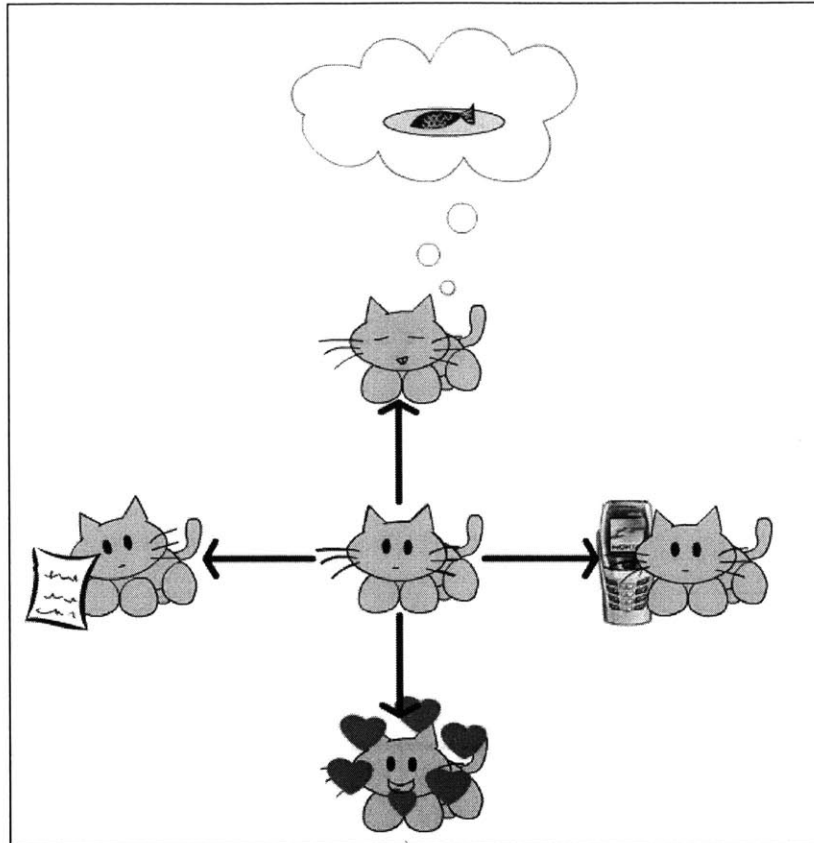


Figure 11: Diagram of possible cat states created with the diagram tool

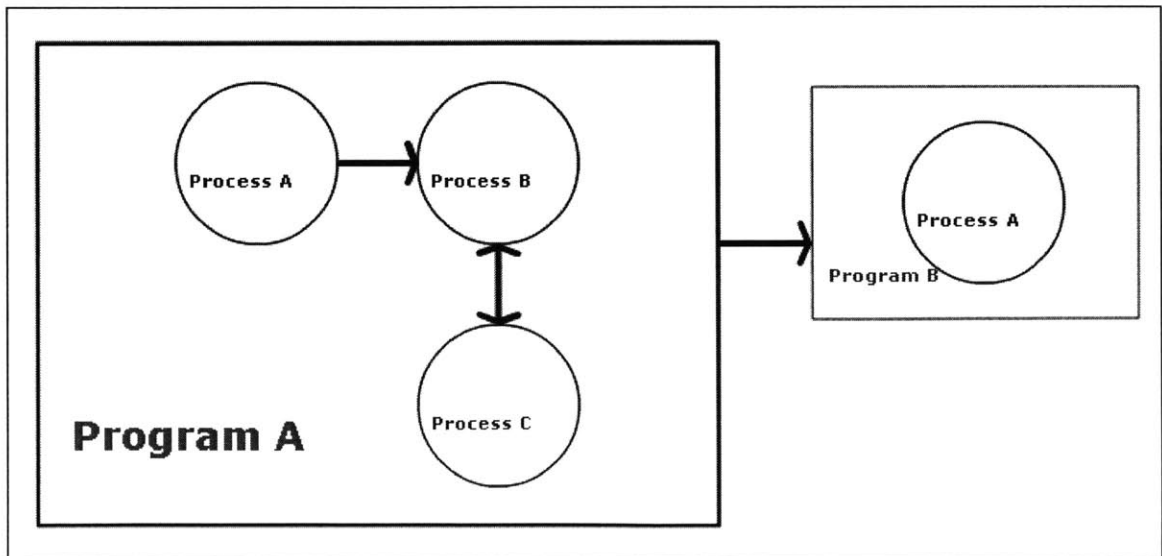


Figure 12: Diagram with composite components

Chapter 9

Conclusion

COMMANIMATION is a speech-controlled animation system that allows users to quickly and easily create and control new and meaningful animations. It is a new approach to animation. This system may not be sufficient for professional programmers who desire very intricate animations. However, this method may act as a fast and powerful tool for creating sketches of animations and laying out the big picture before injecting details.

This research has a number of contributions. This thesis has defined a new approach to animation based on the description of change through commands. Second, the animation commands and features in COMMANIMATION have been expanded. Third, this research work has investigated errors and mechanisms for controlling them. Fourth, this project has evaluated the application of speech to animation. In addition, two additional applications using the COMMANIMATION framework have been created.

As shown through the development and use of COMMANIMATION, speech works well with the new approach of creating animations. First, the new approach of animation based commands and change allows the user to think of animation on a higher level, and hides many of the small details. COMMANIMATION also provides on a library of predefined objects allows the animation to be controlled in real time, eliminating the need to specify timing beforehand. Because this new approach is much less tedious, speech works well. Second, the use of speech naturally assigns an ordering to the spoken commands. Because animation takes place according to a timeline, speech fits animation.

Language is a much richer form of expression that captures many details. The commands, which are expressed as phrases or sentences, are simple, short, and make sense to the user. The nature of these commands allows speech to be used with this system. Speech, in turn, also makes the system feel more natural. The user does not have to think about how to transform the task into a series of mouse clicks and menu traversals, but can directly express his goal in many different ways.

The technology in `COMMANIMATION` introduces a whole new set of errors. The speech software introduces transcription errors. These errors must be handled in a useful way so that the user does not become annoyed or lose interest. In addition, `COMMANIMATION` also runs over three networked computers. In such a complex system, errors are bound to occur. However, the sources of these errors are hard to determine. `COMMANIMATION` provides a platform to study error detection and error control.

A developer can use the `COMMANIMATION` framework to create new applications. Chapter 8 described two new applications, the `StoryBook` and the `Diagram` tool. For the `StoryBook`, the original set of animation commands was sufficient. For the `Diagram` tool, additional primitive functions were implemented to support the new diagram actions. However, the original group of commands is a good core set that can be used to build new functions for new applications.

Although the `StoryBook` and `Diagram` tool are both speech-enabled applications, the `StoryBook` application seems much more impressive or magical than the `Diagram` tool. Perhaps speech feels more connected to storytelling than to creating diagram tools. Or perhaps, storytelling feels like a more effortless task than creating diagrams. In any

case, applying speech control to an animated story book seems to engage the user even more, enhances the task of storytelling, and creates a magical effect for the user.

Bibliography

- [1] Adler, Aaron and Davis, Randall. Speech and sketching for multimodal design. In *Proceedings of Intelligent User Interfaces 2004*, pp. 214-216, 2004.
- [2] Bazzi, I., and Glass, J. Modeling out-of-vocabulary words for robust speech recognition. In *Proceedings of the Sixth International Conference on Spoken Language Processing*, Beijing, China, 2000.
- [3] Borchardt, G. Causal Reconstruction. Tech. Report AIM-1403, Massachusetts Institute of Technology, February 1993.
- [4] Brown, M., Glinski, S., Goldman, B., Schmult, Brian. PhoneBrowser: A Web-Content-Programmable Speech Processing Platform. Position Paper for the W3C Workshop on Voice Browsers, Cambridge, MA, October 1998.
- [5] Coates, A. Speech Controlled Animation. Bachelor of Science Thesis, University of York, Department of Computer Science, Heslington, York, March 2002.
- [6] Conway, M., Audia, S., Burnette, T., Cosgrove, D., Christiansen, K., Deline, R., Durbin, J., Gossweiler, R., Koga, S., Long, C., Mallory, B., Miale, S., Monkaitis, K., Patten, J., Pierce, J., Shochet, J., Staack, D., Stearns, B., Stoakley, R., Sturgill, C., Viega, J., White, J., Williams, G., and Pausch, R. Alice: Lessons Learned from Building a 3D System for Novices. In *Proceedings of CHI*, The Hague Amsterdam, April 2000.
- [7] Coyne, B., and Sproat, R. WordsEye: An Automatic Text to Scene Conversion System. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, 2001.
- [8] Davis, B. LifeLine: Voice Action Adventure. GameSpot Review, <<http://www.gamespot.com/ps2/action/operatorsside/review.html>>, March 2004.
- [9] Deathridge, B. Auditory and Other Sensory Forms of Information Presentation. Human engineering guide to equipment design, Washington D.C.: US Government Printing Office, 1972.
- [10] Edify Corporation. Speech Recognition is the key to Better Customer Service. <http://www.edify.com/pdf/ViecoreSprint_CS.pdf>, April 2004.
- [11] Goose, S., Wynblatt, M., Mollenhauer, H. 1-800-Hypertext: Browsing Hypertext with a Telephone. In *Proceedings of the Ninth ACM Conference on Hypertext and HyperMedia*, Pittsburgh, PA, June 1998.

- [12] Hampicke, M. Smart Home: Speech Based User Interfaces for Smart Home Applications. COST219 Seminar - Speech and Hearing Technology, Cottbus, Germany, November 2000.
- [13] Johnston, M., Bangalore, S., Vasireddy, G., Stent, A., Ehlen, P., Walker, M., Whittaker, S., and Maloor, P. MATCH: An architecture for multimodal dialogue systems. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 376-383, 2002.
- [14] Kim, H. Multimodal Animation Control. Master of Engineering Thesis, Massachusetts Institute of Technology, May 2003.
- [15] Ko, T., Demirdjian, D., and Darrell, T. Untethered Gesture Acquisition and Recognition for a Multimodal Conversational System. In *Proceedings of the 5th International Conference on Multimodal Interface*, pages 147-159, Vancouver, Canada, 2003.
- [16] Kuo, J. An XML Messaging Protocol for Multimodal Galaxy Applications, Master of Engineering Thesis, Massachusetts Institute of Technology, May 2002.
- [17] Lamont, Ian. Speech recognition technology will hear you now. CNN.com, <<http://www.cnn.com/2001/TECH/industry/06/06/auto.speech.idg>>, June 2001.
- [18] Markowitz, J. Driving into the Future. *Speech Technology Magazine*, May/June 2003.
- [19] McDermottroe, J. Speech Controlled Multimedia System. Bachelor of Arts Thesis, Trinity College, Dublin, May 2003.
- [20] Moore, R., Dowding, J., Bratt, H., Gawron, J., Gorf, Y., Cheyer, A. CommandTalk: A spoken-language interface for battlefield simulations. In *Proceedings of the Conference on Applied Natural Language Processing*, pages 1-7, Washington, D.C., April 1997.
- [21] Moskowitz, J. Speech Recognition with Windows XP. <<http://www.microsoft.com/windowsxp/expertzone/columns/moskowitz/02september23.asp>>, September 2002.
- [22] Nigam, A. Analytical Techniques for Debugging Pervasive Computing Environments. Master of Engineering Thesis, Massachusetts Institute of Technology, Cambridge, MA, 2003.
- [23] Seals, C., Rosson, M., Carroll, J., Lewis, T., and Colson, L. Stagecast Creator: an exercise in minimalism and collaboration. In *Proc. IEEE Symp. on Human-Centric Computing 2002*, pages 177-186, Arlington, VA, September 2002.

- [24] Seneff, Stephanie. Response Planning and Generation in the MERCURY Flight Reservation System. *Computer Speech and Language*, 16, pp. 283-312, 2002.
- [25] Tollmar, K., Demirdjian, D., and Darrell, T. Gesture + Play: Exploring Full-Body Navigation for Virtual Environments. In *Proceedings of CVPRHCI*, Madison, WI, June 2003.
- [26] Weinstein, Eugene. SpeechBuilder: Facilitating Spoken Dialogue System Development. Master of Engineering Thesis, Massachusetts Institute of Technology, May 2001.
- [27] Wilson, J. Reporting from the Field: Speech Transforms Data Collectors into Knowledge Workers. *Speech Technology Magazine*, February/March 2000.
- [28] Zongker, D., and Salesin, D. On Creating Animated Presentation. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, San Diego, CA, 2003.
- [29] Zue, V., Seneff, S., Glass, J., Polifroni, J., Pao, C., Hazen, T., Hetherington, Lee. JUPITER: A Telephone-Based Conversational Interface for Weather Information. *IEEE Transactions on Speech and Audio Processing*, Vol. 8, No. 1, January 2000.