P- 1811

ANIMATION OF 3D CHARACTERS WITH SYNCHRONIZED SPEECH

A PROJECT REPORT

Submitted by

SONA K	71203104049
VYSNAVI Y	71203104059

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2007

P- 1811

BONAFIDE CERTIFICATE

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"ANIMATION OF 3D CHARACTERS WITH SYNCHRONIZED SPEECH"** is the bonafide work of "**SONA K (71203104049)** and **VYSNAVI Y (71203104059)**" who carried out the project work under my supervision.

[illegible]	[illegible]
SIGNATURE	**SIGNATURE**
Prof. S. Thangasamy	Mrs. R. Kalaiselvi
HEAD OF THE DEPARTMENT	**SUPERVISOR**
	Senior Lecturer
Department of Computer Science & Engg.	Department of Computer Science & Engg.
Kumaraguru College of Technology, Chinnavedampatti Post, Coimbatore – 641 006	Kumaraguru College of Technology, Chinnavedampatti Post, Coimbatore – 641 006

Submitted for viva-voce examination held on 24.4.2007

[illegible] 24/4	[illegible] 24/4/07
INTERNAL EXAMINER	**EXTERNAL EXAMINER**

DECLARATION

DECLARATION

We hereby declare that the project entitled **"ANIMATION OF 3D CHARACTERS WITH SYNCHRONIZED SPEECH"** is a record of original work done by us and to the best of our knowledge; a similar work has not been submitted to Anna University or any institution, for fulfillment of the requirement of the course study.

The report is submitted in partial fulfillment for the award of the degree of **Bachelor of Computer Science and Engineering** of Anna University, Chennai.

Place : Coimbatore

Date : 18.4.2007

[illegible]
(Sona K)

[illegible]
(Vysnavi Y)

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

We like to extend our sincere and heartfelt thanks to Mrs. R. Kalaiselvi, Senior Lecturer, Department of Computer Science and Engineering for her constant support and guidance extended towards the successful completion of this project.

We are greatly indebted to our beloved Head of the Department, Computer Science and Engineering, Prof. S. Thangasamy and Ms. S. Rajini, Project coordinator for their help and guidance in finishing this project to our fullest satisfaction.

We have great pleasure in extending our gratitude to other teaching staff and lab assistants, Department of Computer Science and Engineering, for their support in this project. We would also like to acknowledge The Department Librarian for her kind cooperation.

We like to thank all our friends and well wishers for all their kind help extended towards the promotion of this project.

ABSTRACT

ABSTRACT

We present a facial model designed primarily to support animated speech. Our facial model takes facial geometry as input and transforms it into a parametric deformable model. The facial model uses a muscle-based parameterization, allowing for easier integration between speech synchrony and facial expressions. It has a highly deformable lip model that is grafted onto the input facial geometry to provide the necessary geometric complexity needed for creating lip shapes and high-quality renderings. It also includes a highly deformable tongue model that can represent the shapes the tongue undergoes during speech. We add teeth, gums, and upper palate geometry to complete the inner mouth. To decrease the processing time, we hierarchically deform the facial surface. We also present a method to animate the facial model over time to create animated speech using a model of co-articulation that blends visemes together using dominance functions. We treat visemes as a dynamic shaping of the vocal tract by describing visemes as curves instead of key frames. We show the utility of the techniques described in this paper by implementing them in a text-to-audiovisual-speech system that creates animation of speech from unrestricted text. The facial and co-articulation models must first be interactively initialized. The system then automatically creates accurate real-time animated speech from the input text. It is capable of cheaply producing tremendous amounts of animated speech with very low resource requirements.

TABLE OF CONTENTS

TABLE OF CONTENTS

INTRODUCTION

1. INTRODUCTION

Synchronizing the motion of the lips with audio is difficult for several reasons. The motion itself is quite complex and the movements are very fast. The parts of the face involved with speech are extremely deformable and complex. The visual portion of speech is affected by not only what is spoken, but also by how it is said. The emotional state of the speaker or the listener, the noise level, the underlying deeper meaning of the utterance (for instance, stressing a word, conveying emotion, etc.), and the state of the conversation are just some of the many things that affect the visual part of speech. This means that, for instance, a word or phrase will rarely look exactly the same when spoken at different times. To exacerbate the problem the audiences are all experts at watching speech. They may not be trained to read lips, but they know how the visual and audial portions of speech align and they are very sensitive to inaccuracies. As well, if the visual portion of the speech is incorrect it could lead to incorrect interpretation.

This research addresses the problem of creating realistic animated speech by tackling both modeling and animating. Without a highly deformable model, the best animation method will fall short. Conversely, without a good animation method, a highly deformable model will not matter. Since the information in the visual part of speech is concentrated in the mouth area, our facial model was built from the mouth outward, starting with deformable lips, a deformable tongue, and full mouth geometry. The facial model takes existing geometry of the skin and adds the deformable mouth, an underlying skull and virtual muscles to control deformation of the mouth as well as the rest of the face. The model is deformed in a hierarchical manner to achieve real-time animation.

To animate facial models many methods have been used to transition between key poses. Linear or spline interpolation which works well for most animation, falls short when animating speech (mostly because they give incorrect accelerations). The primary problem with animating speech is the difficulty of creating the key frames for a given utterance. The common approach is to break the speech into phonemes (phonetic elements) and then convert them into visemes (visual elements) which become the key frames. But a phoneme does not always look the same. Its appearance depends on its emphasis, duration and the neighboring phonemes. These effects are known as co-articulation and prosody.

Co-articulation is the effect a phoneme has on, and is affected by its neighboring phonemes. The vocal tract is made of tissues that have finite acceleration and deceleration. Often there is insufficient time between phonemes for an articulator to reach the next ideal position or to hold it long enough. The articulator may start its motion earlier or lag behind, blurring the lines between phonemes. It may also not be necessary to reach the ideal target for each phoneme in order to get the intended message across, so to conserve energy, the target may be ignored. For some phonemes, articulators may have a very large number of choices for valid target locations. The production of speech is nonlinear and the speaker not only chooses which targets to reach based on the intended message, but also on feedback from the listener. Prosodic features of speech include length, accent, stress, tone and intonation and they affect the timing and position of the vocal tract parts. Co-articulation is physical, while prosody is systemic.

To create realistic motion we develop a co-articulation model that treats visemes as a dynamic shaping of the vocal tract instead of as a single key frame as is commonly done. Our co-articulation model is designed to be used for multiple languages and for varying situations. It is designed to handle the blending of phonemes due to the limited time, velocities and accelerations of the vocal tract parts to reach their ideal locations during conversational speech. Our animation method is separate from our facial model and these techniques can be used to drive other facial models. Models that can be deformed based on muscle action can be driven directly by our viseme sets and those using other methods would require that new viseme sets be defined.

We use our facial model in a text-to-audiovisual-speech (TTAVS) system that can speak or sing in multiple languages using multiple characters to show the utility of the facial model as well as the effectiveness of our co-articulation model. Our TTAVS system has some preprocessing costs, but once set up, can generate novel animated speech, without any user intervention, quickly and accurately, from just text. This is in contrast to the entertainment industry which relies on the talent and time of artists or on the expensive capture of an actor's performance. By using synthetic speech, we keep the cost of generating animation to a minimum. As well, updates require only new text to be created. Synthetic speech currently lacks realism and sounds very emotionless; however active research in speech generation is constantly improving the quality. Because of the significantly lower cost of animation, synthetic speech is the best choice for our application of synthetic conversational agents for human computer interaction.

One contribution of this work is a facial model designed specifically for speech that takes existing geometry of a head and adds separate lip and tongue models, full mouth geometry, a realistically shaped skull with accurate jaw motion, and is driven hierarchically based on muscles. Finally we create an animation system that generates accurate, automatic, speech-synchronized animation from only text. This system can use any face geometry with only some preprocessing costs and generate countless hours of cheap good facial animation. These animations can be easily modified if desired using a human animator for the creative task of adding emotion with the difficult task of speech synchrony done by the system.

LITERATURE REVIEW

2. LITERATURE REVIEW

2.1 PROBLEM DOMAIN

The research addresses the problem of creating realistic animated speech by tackling both modeling and animating. Without a highly deformable model the best animation method will fall short. Conversely, without a good animation method a highly deformable model will not matter. Since the information in the visual part of speech is concentrated in the mouth area, our facial model was built from the mouth outward, starting with deformable lips, a deformable tongue, and full mouth geometry. The facial model takes existing geometry of the skin and adds the deformable mouth, an underlying skull, and virtual muscles to control deformation of the mouth as well as the rest of the face. The model is deformed in a hierarchical manner to achieve real-time animation.

2.2 EXISTING SYSTEM

To animate facial models, many methods have been used to transition between key poses. Linear or spline interpolation which works well for most animation, falls short when animating speech, mostly because they give incorrect accelerations. The primary problem with animating speech is the difficulty of creating the key frames for a given utterance.

Speech animation methods can be divided into three categories: motion capture, image-based methods and 3D methods. Each category has strengths and weaknesses. Some of the best results of speech synchronization involve some sort of motion capture. The resulting speech synchronized animation is of extremely good quality if good motion capture is combined with a high quality facial model. This solution has two main problems, one is the cost in time and equipment. The other is that all needed motion must be recorded beforehand, no novel animations can be generated on-the-fly.

2.2.1 PROBLEMS FACED

- Cost and Equipment
- Video to be recorded before hand

2.3 PROPOSED SYSTEM

The proposed changes that reduce some unnecessary controls, add some extra controls and use it in a different manner, defining viseme curves instead of viseme targets. The result is a co-articulation model just as powerful but with fewer drawbacks. The system desire a technique that is able to take advantage of laser scanners and human modelers.

The geometry produced by these sources can be extremely realistic. However, these models often lack sufficient detail in the mouth area (particularly laser scans), especially the geometry inside the mouth and lips. This missing dètail will cripple animated speech. We graft a lip model to the input geometry to get the benefits of both the full input geometry and a high resolution mouth. Grafting the lip geometry is an interactive process that takes on the order of 15-30 minutes.

The system also adds a deformable tongue, teeth, gums, an upper palate and articulable eyes. Adding the inner mouth geometry and the eyes, results in significantly better animations. The tongue is the most important for improving intelligibility, while eyes that don't remain still have a dramatic impact on giving the animation life. The teeth and other mouth parts have the least impact.

2.3.1 ADVANTAGES

- Cost effective
- Not Time consuming
- Produces very realistic animation

2.4 SCOPE OF THE PROJECT

Facial modeling and animation have diverse uses in a wide range of fields including

- The entertainment industry, education, computer vision, surgery, human-computer interaction, and virtual environments to name a few.
- The entertainment industry, which includes film, television, commercials, theme rides and games have produced some of the most realistic facial animations.
- Use facial animation to tell us a story, make us laugh, make us cry, make us spend our money, and help us immerse ourselves in a new world if only for a little while.
- They use computer graphics to create scenes that would otherwise be impossible or prohibitively expensive to create.
- Examples include scenes of talking Martians, animals, John F. Kennedy, John Wayne, etc.
- They achieve these results at great costs, in talent, equipment and time.

PROGRAMMING ENVIRONMENT

3. PROGRAMMING ENVIRONMENT

3.1 HARDWARE REQUIREMENTS

• Processor Type	:	Intel Pentium IV
• Processor Speed	:	3.8 GHZ
• RAM	:	512 MB RAM
• Hard Disk Capacity	:	40 GB
• Speakers		

3.2 SOFTWARE REQUIREMENTS

• Operating System	:	Windows 2000, XP
• Language	:	Visual Basic .NET 1.1
• Speech Engine	:	Microsoft Speech SDK 5.1
• Modeling Software	:	3D Studio MAX 7.0

3.2.1 INTRODUCTION TO THE VISUAL BASIC .NET 1.1

.NET is going to change the way you design your applications as much as the introduction of classes to VB changed the best way to build your VB5 or 6 applications. With its release for the .NET platform, the Visual Basic language has undergone dramatic changes.

- The language itself is now fully object-oriented.
- Applications and components written in Visual Basic .NET have full access to the .NET Framework, an extensive class library that provides system and application services.
- All applications developed using Visual Basic .NET run within a managed runtime environment, the .NET common language runtime.

Visual Basic .NET is the next generation of Visual Basic, but it is also a significant departure from previous generations. Experienced Visual Basic 6 developers will feel comfortable with Visual Basic .NET code and will recognize most of its constructs. However, Microsoft has made some changes to make Visual Basic .NET a better language and an equal player in the .NET world. These include such additions as a Class keyword for defining classes and an Inherits keyword for object inheritance among others. The Visual Basic .NET compiler without significant modification can't compile Visual Basic 6 code. The good news is that Microsoft has provided a migration tool to handle the task.

ARCHITECTURE OF THE SYSTEM

4. ARCHITECTURE OF THE SYSTEM

The given input text is taken to the Phoneme transcriptor where the corresponding viseme for the word is identified from the speech library. The LPC coefficients determine the pitch and quality of speech produced from the library. The visual images of a human model created earlier are applied for achieving the accuracy. The virtual human image with the different mouth and jaw movements are created to achieve a better result. The synchronization of both audio and visual speech is generated using the "Time delay neural networks algorithm". The Process of synchronization is achieved for any number of text input.

$$\begin{tikzcd}
\text{Accessing appropriate phoneme} \arrow[d] \\
\text{Viseme generation} \arrow[d] \\
\text{Synchronization}
\end{tikzcd}$$

Fig. 4.1 Architectural flow of the system

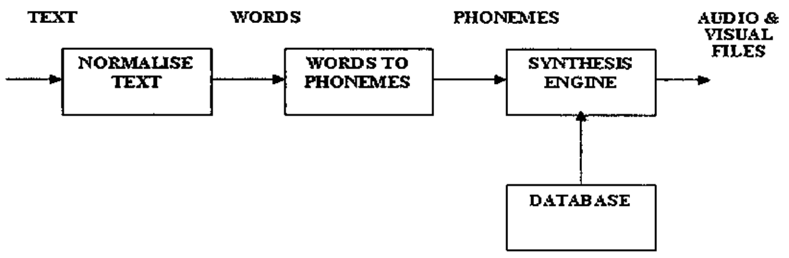


Fig. 4.2 Various processes done in the system

4.1 LIP SYNCHRONIZATION

While lip reading represents the analysis aspect of audiovisual speech processing, the synthesis aspect of audiovisual speech processing is audiovisual speech synthesis. While audio speech synthesis has been well studied by the speech community for many years, visual speech synthesis has attracted much attention in recent years both from the speech community and from the computer graphics community. The goal of visual speech synthesis research is to animate a face model to render lip movements that are synchronized with the acoustic speech signal. Major applications include human-computer interaction, computer-aided instruction, cartoon animation, video games and multimedia telephony for hearing-impaired individuals.

For face modeling, two- or three-dimensional (3-D) meshes are often used. A mesh is composed of a number of triangles (sometimes quadrilaterals) that model the shape of a human face. To animate the model to render various facial expressions, the facial action coding system (FACS) is often used to generate the required movement of these triangles. Parke developed the first 3-D face model in the early 1970s. Most of the 3-D face models currently in use are derivatives of Parke's model. These face models are specified by the coordinates of all the vertices. Animating these models involves explicitly manipulating the vertexes. This method is referred to as direct parameterization. Waters developed a muscle-based face model that simulates the anatomic structure of the human face. In addition to a number of triangles, the model also contains a set of face "muscles." The face model can be animated by stretching and contracting these muscles. Muscle-based face models are typically easier to control since we only have to specify the strength for the muscles, rather than all the vertexes. Massaro and Cohen have one of the most realistic 3-D face models. Their face model has a tongue and teeth and can do sophisticated texture mapping.

Instead of relying on a 3-D mesh as a face model, the work in uses a set of mouth images to create the lip synchronization. Starting from the original video sequence of a talking person, the sequence is long enough to cover all the possible mouth shapes of the person; they analyze the sequence to derive the phonetic transcription. When presented with new input audio, segments from the original video with matching phonemes are extracted and concatenated together with image smoothing to render lip synchronization.

MICROSOFT SPEECH ENGINE

5. MICROSOFT SPEECH ENGINE

Speech SDK 5.0 is the software speech synthesizer developed by Microsoft Corporation. The Microsoft Speech API provides applications with the ability to incorporate speech recognition (command & control or dictation) or text-to-speech using either C/C++ or Visual Basic. SAPI follows the OLE Component Object Model (COM) architecture. It is supported by many major speech technology vendors. The major interfaces are

- **Voice Commands:**

  A high level speech recognition API for command and control.

- **Voice Text:**

  A simple high level text-to-speech API.

- **Speech Recognition:**

  It provides detailed control of a speech recognition engine for both command-and-control and dictation.

- **Text-to-Speech:**

  It provides detailed interface to a text-to-speech engine for control of playback, speaking style, voice quality etc.

- **Multimedia Audio Objects:**

  The audio I/O for microphones, headphones, speakers, telephone lines, files etc.

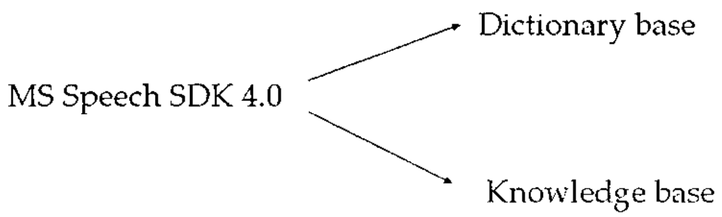


Fig. 5.1 Accessing appropriate phoneme

5.1 SPEECH SYNTHESIS

Speech synthesis is the artificial production of human speech. A computer system used for this purpose is called a speech synthesizer and can be implemented in software or hardware. A text-to-speech (TTS) system converts normal language text into speech; other systems render symbolic linguistic representations like phonetic transcriptions into speech

Synthesized speech can also be created by concatenating pieces of recorded speech that are stored in a database. Systems differ in the size of the stored speech units; a system that stores phones or diphones provides the largest output range but may lack clarity. For specific usage domains, the storage of entire words or sentences allows for high-quality output. Alternatively, a synthesizer can incorporate a model of the vocal tract and other human voice characteristics to create a completely synthetic voice output.

The quality of a speech synthesizer is judged by its similarity to the human voice, and by its ability to be understood. An intelligible text-to-speech program allows people with visual impairments or reading disabilities to listen to written works on a home computer. Many computer operating systems have included speech synthesizers since the early 1980s.

5.2 TEXT PROCESSING

A text-to-speech system is composed of two parts: a front-end and a back-end. The front-end has two major tasks. First it converts raw text containing symbols like numbers and abbreviations into the equivalent of written-out words. This process is often called text normalization, pre-processing, or tokenization. The front-end then assigns phonetic transcriptions to each word, and divides and marks the text into prosodic units, like phrases, clauses, and sentences. The process of assigning phonetic transcriptions to words is called text-to-phoneme or grapheme-to-phoneme conversion. Phonetic transcriptions and prosody information together make up the symbolic linguistic representation that is output by the front-end. The back-end—often referred to as the synthesizer—then converts the symbolic linguistic representation into sound.

Text-to-phoneme challenges

Speech synthesis systems use two basic approaches to determine the pronunciation of a word based on its spelling, a process which is often called text-to-phoneme or grapheme-to-phoneme conversion (phoneme is the term used by linguists to describe distinctive sounds in a language). The simplest approach to text-to-phoneme conversion is the dictionary-based approach, where a large dictionary containing all the words of a language and their correct pronunciations is stored by the program. Determining the correct pronunciation of each word is a matter of looking up each word in the dictionary and replacing the spelling with the pronunciation specified in the dictionary. The other approach is rule-based, in which pronunciation rules are applied to words to determine their pronunciations based on their spellings. This is similar to the "sounding out" or synthetic phonics, approach to learning reading.

Each approach has advantages and drawbacks. The dictionary-based approach is quick and accurate, but completely fails if it is given a word which is not in its dictionary. As dictionary size grows, so too does the memory space requirements of the synthesis system. On the other hand, the rule-based approach works on any input, but the complexity of the rules grows substantially as the system takes into account irregular spellings or pronunciations. (Consider that the word "of" is very common in English, yet is the only word in which the letter "f" is pronounced [v].) As a result, nearly all speech synthesis systems use a combination of these approaches.

Some languages, like Spanish, have a very regular writing system, and the prediction of the pronunciation of words based on their spellings is quite successful. Speech synthesis systems for such languages often use the rule-based method extensively, resorting to dictionaries only for those few words, like foreign names and borrowings, whose pronunciations are not obvious from their spellings. On the other hand, speech synthesis systems for languages like English which have extremely irregular spelling systems, are more likely to rely on dictionaries and to use rule-based methods only for unusual words or words that are not in their dictionaries.

6. 3D STUDIO MAX 7.0

3ds Max is a full-featured 3D graphics application developed by AutoDesk Media and Entertainment. It runs on the Win32 and Win64 platforms. As of August 2006, 3ds Max is in its ninth version generation.

The original 3D Studio product was created for the DOS platform by the Yost Group and published by AutoDesk. AutoDesk purchased the product at its second release mark and internalized development entirely over the next two releases. After 3D Studio Release 4, the product was ported to the Windows NT platform and originally named "3D Studio MAX". This version was also originally created by the Yost Group. It was released by Kinetix, which was at that time AutoDesk's division of media and entertainment. Later the product name was changed to "3ds max" (all lower case) to better comply with the naming conventions of Discreet, a Montreal-based software company which AutoDesk had purchased. At release 8, the product was again branded with the AutoDesk logo and the name was again changed to "3ds Max" (upper and lower case).

6.1 OVERVIEW

3ds Max is one of the most widely-used off the shelf 3D animation programs by the general public. It has strong modeling capabilities, a flexible plug-in architecture and a long heritage on the Microsoft Windows platform. It is mostly used by video game developers, TV commercial studios and architectural visualization studios. It is also used for movie effects and movie pre-visualization.

In addition to its modeling and animation tools, the latest version of 3ds Max also features advanced shaders (such as ambient occlusion and subsurface scattering), dynamic simulation, particle systems, radiosity, normal map creation and rendering, global illumination, an intuitive and fully-customizable user interface, its own scripting language and much more. There is also a plethora of specialized renderer plug-in that can be bought separately, such as V-Ray, Brazil r/s and final Render.

Earlier versions required a special copy prevention device (a dongle) to be plugged into the parallel port while the program was run, but later a software copy prevention method was implemented instead. Registration involving personal information such as name, address and e-mail address is now required.

Table 1. Viseme Groups for English			
1	**p,b,m**	**8**	**n,l**
2	**f,v**	**9**	**r**
3	**th,dh**	**10**	**A**
4	**t,d**	**11**	**E**
5	**k,g**	**12**	**I**
6	**sh,zh**	**13**	**O**
7	**s,z**	**14**	**U**

Fig. 6.1 Viseme groups for English language

6.2 FACIAL MODELING

To reach our goal of animated speech, we create a facial model specifically designed for animated speech with a deformable mouth. This is in contrast to most facial models, which are designed for facial expressions. We discuss here the problem of speech-synchronized animation, which we believe is a harder problem. Facial expressions are created using standard techniques. We use a muscle-based parameterization because muscle movement drives the deformation of the face. Higher-level parameterizations that are easier to use by an animator can be defined in terms of these low-level parameters.

Fig. 6.2 Various possible grafted lip models

We desire a technique that is able to take advantage of laser scanners and human modelers. The geometry produced by these sources can be extremely realistic. However, these models often lack sufficient detail in the mouth area (particularly laser scans), especially the geometry inside the mouth (inner lips, teeth, tongue, gums, upper palate). This missing detail will cripple animated speech. We graft a lip model to the input geometry to get the benefits of both the full input geometry and a high resolution mouth. Grafting the lip geometry is an interactive process that takes on the order of 15-30 minutes. We also add a deformable tongue [39], teeth, gums, an upper palate, and articulable eyes. Adding the inner mouth geometry and the eyes, results in significantly better animations.

SYNCHRONIZATION ALGORITHM

7. SYNCHRONIZATION ALGORITHM

7.1 TIME DELAY NEURAL NETWORKS ALGORITHM

Space and time coordinate the physical world that surrounds us. Physical objects exist at some space time point. Such objects may be idle or active, and their forms or behaviors may vary over time. Despite these distortions, people can inherently recognize the objects. To construct an ideal recognizer capable of dealing with natural patterns in daily life, e.g., speech, image, or motion, the recognizer should remain insensitive to the patterns' distortions in the time or space domain, or both. Some criteria are available to assess the recognizer's tolerance for distortions of input patterns. For instance, the translation-invariant property of a recognizer implies that the recognizer can accurately recognize an object regardless of its proximity.

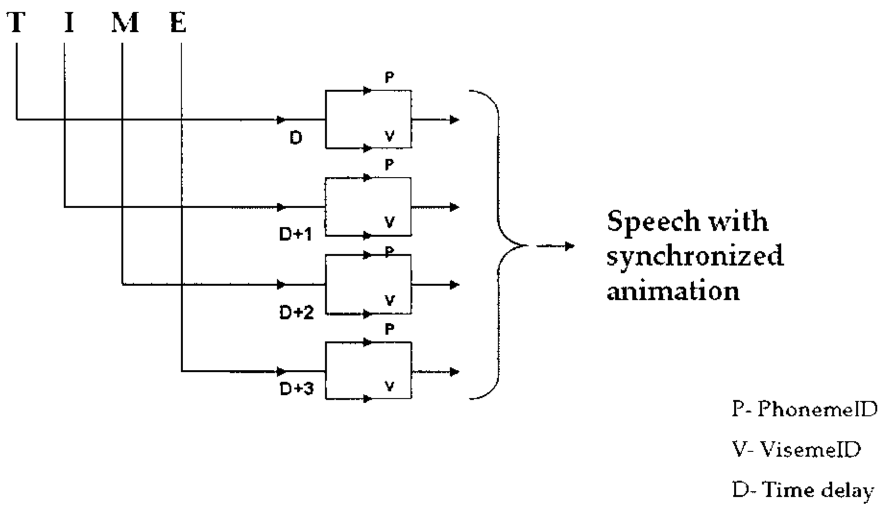


Fig. 7.1 Feed Forward networks for the algorithm

Neural networks are adopted in the developed system herein, since their powerful learning ability and flexibility have been demonstrated in many applications. To achieve this goal, a model must be developed, capable of treating the space-time 3-D dynamic information. Restated, the model should be capable of learning the 3-D dynamic mapping that maps the input pattern varying in the space-time 3-D domain to a desired class. However, according to our results, the conventional neural network structure cannot adequately resolve this problem. The earlier MLP can learn the nonlinear static mapping between the input set and the output set. The inventions of the TDNN and RNN bring the neural network's applications into the spatiotemporal domain, in which the 1-D dynamic mapping between the input set and the output set can be learned. The SDNN, which is evolved from the TDNN, further enhances the ability of the neural networks to learn 2-D dynamic mapping. The related development is curtailed since previous literature has not clarified the need for such models. A more important reason is that the ordinary constituents which are used in MLP, TDNN and SDNN are difficult in terms of constructing the complex network that can represent and learn the 3-D or higher dimensional dynamic information.

TESTING

8. TESTING

Testing is the process of executing a program with the intent of finding errors. Testing can be separated into Verification and Validation testing. This phase involves:

1. Generation of test constraints.
2. Generation of actual test data for satisfying these constraints.
3. Test execution.
4. Output verification.

8.1 UNIT TESTING

This is the lowest level of testing a product. It involves individually testing each small unit of code to ensure that it works on its own, independent of the other units. It tells the developers that an application's pieces are working as designed. The developer establishes the test cases, test procedure and test data for testing the software corresponding to each software unit.

Each module of our project were executed separately and checked for correct working of the modules. The intended outputs for the algorithm were thus obtained after the unit testing.

8.2 VERIFICATION TESTING

Verification methods are used to ensure the system (software, hardware) complies with an organization standard and processes relying on review. In our system modules were tested for correct functionality.

8.3 VALIDATION TESTING

Validation ensures that a system operates according to the plans by executing real-life function. Validation is done for modules. This test is done to check the validity of the entire input.

8.4 FUNCTIONAL TESTING

Functional testing is a Black Box testing because of no knowledge. They verify all validation tests and inspect how the system performs. Each module is tested for expected output, without tracing the path of code. This testing methodology views each program as a mathematical function that maps its inputs to its outputs. It uncovers any errors that occur in implementing requirements or in design specifications. These outputs of the reports were verified for correctness and completeness.

8.5 STRUCTURAL TESTING

Structural Testing is called White Box Testing because the internal logic of the system is known. The purpose of this testing is to check out whether the logical operation of the system works well. Here the processing of the input is considered.

8.6 INTEGRATION TESTING

The purpose of this integration testing is to test the project as a whole after combining the individual modules. Here, the tested modules are combined into sub-systems, which are then tested. This is done to test whether the modules can be integrated properly, emphasizing on interface between modules.

The software was subjected to Integration testing and the different modules were linked together and executed.

The verification and validation of a software system is an abiding process through each phase of the software development process. Testing plays a vital role in determining the reliability and efficiency of the software and fro this reason is a very imperative stage in software development.

FUTURE ENHANCEMENTS

9. FUTURE ENHANCEMENTS

Creating realistic intelligible animated speech requires a deformable mouth, complete with tongue and teeth. As well, interpolating between key shapes is inadequate since it causes unrealistic velocities and accelerations and it ignores co-articulation effects which are quite important. The best is a co-articulation model that approximates the targets, but has the ability to interpolate when needed. The co-articulation model that we present here is an improvement over existing models and can be used with any facial animation system. The viseme sets we have developed should also be adaptable as they are muscle-based and give an intensity that can be mapped to many systems. The modeling techniques can also be used with other systems, especially the individual lip or tongue models. Combining our method of animating speech with other 3D methods of animating expressions should also be straightforward. For an automated speech system that generates the visuals from only text input, we feel our system produces very strong results.

Lip animation suffers terribly from aliasing effects. The lips often move quite fast and may only be at a target location for a very short period. For instance, consider a fast /p/. The lips may only be in contact for less than a frame. The contact could then be missed. Using a complex selection method, the frames could be chosen with uneven intervals to reduce these effects. A better solution is motion blur, but this is expensive.

We also need to make some advances on the modeling side. Our current focus has been on the motion of the lips due to muscle contractions; however, we also need to consider deformations due to collisions between the lips and other parts of the face. The lips must flow around the teeth and not penetrate them. Furthermore, when the tongue presses against the lips for creating sounds or when wetting them, a slight deformation is needed to improve realism. Finally, when the upper and lower lips come into contact, there are subtle changes that need to be shown.

A common approach, and the one we currently use, is to use phonemes as the smallest unit of speech. Instead of using phonemes, syllables may be a better choice. Syllables should look more similar under different circumstances than phonemes do. However, there are many more syllables (on the order of thousands) than phonemes, so defining the set of syllables is much more costly.

CONCLUSION

10. CONCLUSION

The system “Animation of 3D characters with synchronized speech” is implemented with the help of the Microsoft Speech SDK 5.0 and the modeling software 3D Studio MAX 7.0. This system is created to produce realistic animations with the synchronized lip movements appropriate to the speech.

A proper software development scheme is employed in developing the application and the system is tested for various inputs. The system also supports non dictionary English words. All the exceptions that occurred during the implementation are handled as required.

The performance of the devised system is compared with the existing system and its efficiency is found to be much better.

APPENDIX

11. APPENDIX

11.1 SAMPLE CODE

```
Option Strict Off
Option Explicit On
Friend Class AnimateFace
   Inherits System.Windows.Forms.Form
#Region "Windows Form Designer generated code "
   Public Sub New()
      MyBase.New()
      If m_vb6FormDefInstance Is Nothing Then
         If m_InitializingDefInstance Then
            m_vb6FormDefInstance = Me
         Else
            Try
               'For the start-up form, the first instance created is the default
instance.
               If
System.Reflection.Assembly.GetExecutingAssembly.EntryPoint.DeclaringT
ype Is Me.GetType Then
                  m_vb6FormDefInstance = Me
               End If
            Catch
            End Try
         End If
      End If
      'This call is required by the Windows Form Designer.
      InitializeComponent()
   End Sub
   'Form overrides dispose to clean up the component list.
   Protected Overloads Overrides Sub Dispose(ByVal Disposing As
Boolean)
      If Disposing Then
         If Not components Is Nothing Then
            components.Dispose()
         End If
      End If
      MyBase.Dispose(Disposing)
```

```
    End Sub
    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer
    Public ToolTip1 As System.Windows.Forms.ToolTip
    Public WithEvents VisemePicture As
System.Windows.Forms.PictureBox
    Public WithEvents chkShowEvents As
System.Windows.Forms.CheckBox
    Public WithEvents StopBtn As System.Windows.Forms.Button
    Public WithEvents chkSpFlagNLPSpeakPunc As
System.Windows.Forms.CheckBox
    Public WithEvents chkSpFlagAync As
System.Windows.Forms.CheckBox
    Public WithEvents Frame1 As System.Windows.Forms.GroupBox
    Public WithEvents AudioOutputCB As
System.Windows.Forms.ComboBox
    Public WithEvents MainTxtBox As System.Windows.Forms.TextBox
    Public WithEvents ResetBtn As System.Windows.Forms.Button
    Public WithEvents MouthImgList As AxMSComctlLib.AxImageList
    Public WithEvents DebugTxtBox As System.Windows.Forms.TextBox
    Public WithEvents ComDlg As AxMSComDlg.AxCommonDialog
    Public WithEvents FormatCB As System.Windows.Forms.ComboBox
    Public WithEvents RateSldr As AxMSComctlLib.AxSlider
    Public WithEvents VoiceCB As System.Windows.Forms.ComboBox
    Public WithEvents PauseBtn As System.Windows.Forms.Button
    Public WithEvents SpeakBtn As System.Windows.Forms.Button
    Public WithEvents VolumeSldr As AxMSComctlLib.AxSlider
    Public WithEvents Label1 As System.Windows.Forms.Label
    Public WithEvents Label5 As System.Windows.Forms.Label
    Public WithEvents Label4 As System.Windows.Forms.Label
    Public WithEvents Label3 As System.Windows.Forms.Label
    Public WithEvents Label2 As System.Windows.Forms.Label
    Public WithEvents menuFileSpeakWave As
System.Windows.Forms.MenuItem
    Public WithEvents menuFileSaveToWave As
System.Windows.Forms.MenuItem
    Public WithEvents menuSep As System.Windows.Forms.MenuItem
    Public WithEvents menuFileExit As System.Windows.Forms.MenuItem
    Public WithEvents menuFile As System.Windows.Forms.MenuItem
    Public WithEvents menuAbout As System.Windows.Forms.MenuItem
```

```
   Public WithEvents menuHelp As System.Windows.Forms.MenuItem
   Public MainMenu1 As System.Windows.Forms.MainMenu
   'NOTE: The following procedure is required by the Windows Form
Designer
   'It can be modified using the Windows Form Designer.
   'Do not modify it using the code editor.
   <System.Diagnostics.DebuggerStepThrough()> Private Sub
InitializeComponent()
       Me.components = New System.ComponentModel.Container
       Dim resources As System.Resources.ResourceManager = New
System.Resources.ResourceManager(GetType(AnimateFace))
       Me.ToolTip1 = New
System.Windows.Forms.ToolTip(Me.components)
       Me.VisemePicture = New System.Windows.Forms.PictureBox
       Me.chkShowEvents = New System.Windows.Forms.CheckBox
       Me.StopBtn = New System.Windows.Forms.Button
       Me.chkSpFlagNLPSpeakPunc = New
System.Windows.Forms.CheckBox
       Me.chkSpFlagAync = New System.Windows.Forms.CheckBox
       Me.Frame1 = New System.Windows.Forms.GroupBox
       Me.AudioOutputCB = New System.Windows.Forms.ComboBox
       Me.MainTxtBox = New System.Windows.Forms.TextBox
       Me.ResetBtn = New System.Windows.Forms.Button
       Me.MouthImgList = New AxMSComctlLib.AxImageList
       Me.DebugTxtBox = New System.Windows.Forms.TextBox
       Me.ComDlg = New AxMSComDlg.AxCommonDialog
       Me.FormatCB = New System.Windows.Forms.ComboBox
       Me.RateSldr = New AxMSComctlLib.AxSlider
       Me.VoiceCB = New System.Windows.Forms.ComboBox
       Me.PauseBtn = New System.Windows.Forms.Button
       Me.SpeakBtn = New System.Windows.Forms.Button
       Me.VolumeSldr = New AxMSComctlLib.AxSlider
       Me.Label1 = New System.Windows.Forms.Label
       Me.Label5 = New System.Windows.Forms.Label
       Me.Label4 = New System.Windows.Forms.Label
       Me.Label3 = New System.Windows.Forms.Label
       Me.Label2 = New System.Windows.Forms.Label
       Me.MainMenu1 = New System.Windows.Forms.MainMenu
       Me.menuFile = New System.Windows.Forms.MenuItem
       Me.menuFileSpeakWave = New System.Windows.Forms.MenuItem
```

```
        Me.menuFileSaveToWave = New System.Windows.Forms.MenuItem
        Me.menuSep = New System.Windows.Forms.MenuItem
        Me.menuFileExit = New System.Windows.Forms.MenuItem
        Me.menuHelp = New System.Windows.Forms.MenuItem
        Me.menuAbout = New System.Windows.Forms.MenuItem
        Me.Frame1.SuspendLayout()
        CType(Me.MouthImgList,
System.ComponentModel.ISupportInitialize).BeginInit()
        CType(Me.ComDlg,
System.ComponentModel.ISupportInitialize).BeginInit()
        CType(Me.RateSldr,
System.ComponentModel.ISupportInitialize).BeginInit()
        CType(Me.VolumeSldr,
System.ComponentModel.ISupportInitialize).BeginInit()
        Me.SuspendLayout()
        '
        'VisemePicture
        '
        Me.VisemePicture.BackColor =
System.Drawing.SystemColors.Window
        Me.VisemePicture.BorderStyle =
System.Windows.Forms.BorderStyle.Fixed3D
        Me.VisemePicture.Cursor = System.Windows.Forms.Cursors.Default
        Me.VisemePicture.Font = New System.Drawing.Font("Arial", 8.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
        Me.VisemePicture.ForeColor =
System.Drawing.SystemColors.ControlText
        Me.VisemePicture.Location = New System.Drawing.Point(240, 8)
        Me.VisemePicture.Name = "VisemePicture"
        Me.VisemePicture.RightToLeft =
System.Windows.Forms.RightToLeft.No
        Me.VisemePicture.Size = New System.Drawing.Size(225, 169)
        Me.VisemePicture.TabIndex = 27
        Me.VisemePicture.TabStop = False
        '
        'chkShowEvents
        '
        Me.chkShowEvents.BackColor = System.Drawing.Color.Transparent
        Me.chkShowEvents.Cursor = System.Windows.Forms.Cursors.Default
        Me.chkShowEvents.Font = New System.Drawing.Font("Arial", 8.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
        Me.chkShowEvents.ForeColor =
System.Drawing.SystemColors.ControlText
        Me.chkShowEvents.Location = New System.Drawing.Point(40, 72)
        Me.chkShowEvents.Name = "chkShowEvents"
        Me.chkShowEvents.RightToLeft =
System.Windows.Forms.RightToLeft.No
        Me.chkShowEvents.Size = New System.Drawing.Size(89, 13)
        Me.chkShowEvents.TabIndex = 25
        Me.chkShowEvents.Text = "Show Events"
        '
        'StopBtn
        '
        Me.StopBtn.BackColor = System.Drawing.Color.Transparent
        Me.StopBtn.Cursor = System.Windows.Forms.Cursors.Default
        Me.StopBtn.Enabled = False
        Me.StopBtn.Font = New System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
        Me.StopBtn.ForeColor = System.Drawing.SystemColors.ControlText
        Me.StopBtn.Location = New System.Drawing.Point(584, 56)
        Me.StopBtn.Name = "StopBtn"
        Me.StopBtn.RightToLeft = System.Windows.Forms.RightToLeft.No
        Me.StopBtn.Size = New System.Drawing.Size(96, 32)
        Me.StopBtn.TabIndex = 2
        Me.StopBtn.Text = "Stop"
        '
        'chkSpFlagNLPSpeakPunc
        '
        Me.chkSpFlagNLPSpeakPunc.BackColor =
System.Drawing.Color.White
        Me.chkSpFlagNLPSpeakPunc.Cursor =
System.Windows.Forms.Cursors.Default
        Me.chkSpFlagNLPSpeakPunc.Font = New
System.Drawing.Font("Arial", 8.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.chkSpFlagNLPSpeakPunc.ForeColor =
System.Drawing.SystemColors.ControlText
```

```
    Me.chkSpFlagNLPSpeakPunc.Location = New
System.Drawing.Point(40, 48)
    Me.chkSpFlagNLPSpeakPunc.Name = "chkSpFlagNLPSpeakPunc"
    Me.chkSpFlagNLPSpeakPunc.RightToLeft =
System.Windows.Forms.RightToLeft.No
    Me.chkSpFlagNLPSpeakPunc.Size = New System.Drawing.Size(105,
17)
    Me.chkSpFlagNLPSpeakPunc.TabIndex = 24
    Me.chkSpFlagNLPSpeakPunc.Text = "NLPSpeakPunc"
    '
    'chkSpFlagAync
    '
    Me.chkSpFlagAync.BackColor = System.Drawing.Color.White
    Me.chkSpFlagAync.Cursor = System.Windows.Forms.Cursors.Default
    Me.chkSpFlagAync.Font = New System.Drawing.Font("Arial", 8.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
    Me.chkSpFlagAync.ForeColor =
System.Drawing.SystemColors.ControlText
    Me.chkSpFlagAync.Location = New System.Drawing.Point(40, 24)
    Me.chkSpFlagAync.Name = "chkSpFlagAync"
    Me.chkSpFlagAync.RightToLeft =
System.Windows.Forms.RightToLeft.No
    Me.chkSpFlagAync.Size = New System.Drawing.Size(105, 17)
    Me.chkSpFlagAync.TabIndex = 20
    Me.chkSpFlagAync.Text = "FlagsAsync"
    '
    'Frame1
    '
    Me.Frame1.BackColor = System.Drawing.Color.Transparent
    Me.Frame1.Controls.Add(Me.chkSpFlagAync)
    Me.Frame1.Controls.Add(Me.chkSpFlagNLPSpeakPunc)
    Me.Frame1.Controls.Add(Me.chkShowEvents)
    Me.Frame1.Font = New System.Drawing.Font("Arial", 8.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
    Me.Frame1.ForeColor = System.Drawing.SystemColors.ControlText
    Me.Frame1.Location = New System.Drawing.Point(432, 272)
    Me.Frame1.Name = "Frame1"
    Me.Frame1.RightToLeft = System.Windows.Forms.RightToLeft.No
    Me.Frame1.Size = New System.Drawing.Size(184, 105)
    Me.Frame1.TabIndex = 18
    Me.Frame1.TabStop = False
    Me.Frame1.Text = "Speak Flags"
    '
    'AudioOutputCB
    '
    Me.AudioOutputCB.BackColor =
System.Drawing.SystemColors.Window
    Me.AudioOutputCB.Cursor = System.Windows.Forms.Cursors.Default
    Me.AudioOutputCB.DropDownStyle =
System.Windows.Forms.ComboBoxStyle.DropDownList
    Me.AudioOutputCB.Font = New System.Drawing.Font("Arial", 8.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
    Me.AudioOutputCB.ForeColor =
System.Drawing.SystemColors.WindowText
    Me.AudioOutputCB.Location = New System.Drawing.Point(192, 408)
    Me.AudioOutputCB.Name = "AudioOutputCB"
    Me.AudioOutputCB.RightToLeft =
System.Windows.Forms.RightToLeft.No
    Me.AudioOutputCB.Size = New System.Drawing.Size(220, 22)
    Me.AudioOutputCB.TabIndex = 17
    '
    'MainTxtBox
    '
    Me.MainTxtBox.AcceptsReturn = True
    Me.MainTxtBox.AutoSize = False
    Me.MainTxtBox.BackColor = System.Drawing.SystemColors.Window
    Me.MainTxtBox.Cursor = System.Windows.Forms.Cursors.IBeam
    Me.MainTxtBox.Font = New System.Drawing.Font("Arial", 8.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
    Me.MainTxtBox.ForeColor =
System.Drawing.SystemColors.WindowText
    Me.MainTxtBox.HideSelection = False
    Me.MainTxtBox.Location = New System.Drawing.Point(40, 184)
    Me.MainTxtBox.MaxLength = 0
    Me.MainTxtBox.Multiline = True
    Me.MainTxtBox.Name = "MainTxtBox"
```

```
        Me.MainTxtBox.RightToLeft =
System.Windows.Forms.RightToLeft.No
        Me.MainTxtBox.ScrollBars =
System.Windows.Forms.ScrollBars.Vertical
        Me.MainTxtBox.Size = New System.Drawing.Size(680, 80)
        Me.MainTxtBox.TabIndex = 0
        Me.MainTxtBox.Text = "Enter text you wish spoken here."
        '
        'ResetBtn
        '
        Me.ResetBtn.BackColor = System.Drawing.Color.Transparent
        Me.ResetBtn.Cursor = System.Windows.Forms.Cursors.Default
        Me.ResetBtn.Font = New System.Drawing.Font("Arial", 8.0!,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
        Me.ResetBtn.ForeColor = System.Drawing.SystemColors.ControlText
        Me.ResetBtn.Location = New System.Drawing.Point(584, 88)
        Me.ResetBtn.Name = "ResetBtn"
        Me.ResetBtn.RightToLeft = System.Windows.Forms.RightToLeft.No
        Me.ResetBtn.Size = New System.Drawing.Size(96, 32)
        Me.ResetBtn.TabIndex = 7
        Me.ResetBtn.Text = "Reset"
        '
        'MouthImgList
        '
        Me.MouthImgList.Enabled = True
        Me.MouthImgList.Location = New System.Drawing.Point(600, 376)
        Me.MouthImgList.Name = "MouthImgList"
        Me.MouthImgList.OcxState =
CType(resources.GetObject("MouthImgList.OcxState"),
System.Windows.Forms.AxHost.State)
        Me.MouthImgList.Size = New System.Drawing.Size(38, 38)
        Me.MouthImgList.TabIndex = 28
        '
        'DebugTxtBox
        '
        Me.DebugTxtBox.AcceptsReturn = True
        Me.DebugTxtBox.AutoSize = False
        Me.DebugTxtBox.BackColor =
System.Drawing.SystemColors.Window
```

```
        Me.DebugTxtBox.Cursor = System.Windows.Forms.Cursors.IBeam
        Me.DebugTxtBox.Font = New System.Drawing.Font("Arial", 8.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
        Me.DebugTxtBox.ForeColor =
System.Drawing.SystemColors.WindowText
        Me.DebugTxtBox.Location = New System.Drawing.Point(64, 432)
        Me.DebugTxtBox.MaxLength = 0
        Me.DebugTxtBox.Multiline = True
        Me.DebugTxtBox.Name = "DebugTxtBox"
        Me.DebugTxtBox.ReadOnly = True
        Me.DebugTxtBox.RightToLeft =
System.Windows.Forms.RightToLeft.No
        Me.DebugTxtBox.ScrollBars =
System.Windows.Forms.ScrollBars.Both
        Me.DebugTxtBox.Size = New System.Drawing.Size(640, 96)
        Me.DebugTxtBox.TabIndex = 26
        Me.DebugTxtBox.Text = ""
        Me.DebugTxtBox.WordWrap = False
        '
        'ComDlg
        '
        Me.ComDlg.Enabled = True
        Me.ComDlg.Location = New System.Drawing.Point(648, 384)
        Me.ComDlg.Name = "ComDlg"
        Me.ComDlg.OcxState =
CType(resources.GetObject("ComDlg.OcxState"),
System.Windows.Forms.AxHost.State)
        Me.ComDlg.Size = New System.Drawing.Size(32, 32)
        Me.ComDlg.TabIndex = 29
        '
        'FormatCB
        '
        Me.FormatCB.BackColor = System.Drawing.SystemColors.Window
        Me.FormatCB.Cursor = System.Windows.Forms.Cursors.Default
        Me.FormatCB.DropDownStyle =
System.Windows.Forms.ComboBoxStyle.DropDownList
        Me.FormatCB.Font = New System.Drawing.Font("Arial", 8.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
```

```
        Me.FormatCB.ForeColor =
System.Drawing.SystemColors.WindowText
        Me.FormatCB.Location = New System.Drawing.Point(192, 312)
        Me.FormatCB.Name = "FormatCB"
        Me.FormatCB.RightToLeft = System.Windows.Forms.RightToLeft.No
        Me.FormatCB.Size = New System.Drawing.Size(220, 22)
        Me.FormatCB.TabIndex = 15
        '
        'RateSldr
        '
        Me.RateSldr.Location = New System.Drawing.Point(192, 344)
        Me.RateSldr.Name = "RateSldr"
        Me.RateSldr.OcxState =
CType(resources.GetObject("RateSldr.OcxState"),
System.Windows.Forms.AxHost.State)
        Me.RateSldr.Size = New System.Drawing.Size(220, 16)
        Me.RateSldr.TabIndex = 11
        '
        'VoiceCB
        '
        Me.VoiceCB.BackColor = System.Drawing.SystemColors.Window
        Me.VoiceCB.Cursor = System.Windows.Forms.Cursors.Default
        Me.VoiceCB.DropDownStyle =
System.Windows.Forms.ComboBoxStyle.DropDownList
        Me.VoiceCB.Font = New System.Drawing.Font("Arial", 8.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
        Me.VoiceCB.ForeColor = System.Drawing.SystemColors.WindowText
        Me.VoiceCB.Location = New System.Drawing.Point(192, 272)
        Me.VoiceCB.Name = "VoiceCB"
        Me.VoiceCB.RightToLeft = System.Windows.Forms.RightToLeft.No
        Me.VoiceCB.Size = New System.Drawing.Size(220, 22)
        Me.VoiceCB.TabIndex = 9
        '
        'PauseBtn
        '
        Me.PauseBtn.BackColor = System.Drawing.Color.Transparent
        Me.PauseBtn.Cursor = System.Windows.Forms.Cursors.Default
        Me.PauseBtn.Enabled = False
        Me.PauseBtn.Font = New System.Drawing.Font("Arial", 8.0!,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
        Me.PauseBtn.ForeColor = System.Drawing.SystemColors.ControlText
        Me.PauseBtn.Location = New System.Drawing.Point(488, 88)
        Me.PauseBtn.Name = "PauseBtn"
        Me.PauseBtn.RightToLeft = System.Windows.Forms.RightToLeft.No
        Me.PauseBtn.Size = New System.Drawing.Size(96, 32)
        Me.PauseBtn.TabIndex = 3
        Me.PauseBtn.Text = "Pause"
        '
        'SpeakBtn
        '
        Me.SpeakBtn.BackColor = System.Drawing.Color.Transparent
        Me.SpeakBtn.Cursor = System.Windows.Forms.Cursors.Default
        Me.SpeakBtn.Font = New System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
        Me.SpeakBtn.ForeColor = System.Drawing.SystemColors.ControlText
        Me.SpeakBtn.Location = New System.Drawing.Point(488, 56)
        Me.SpeakBtn.Name = "SpeakBtn"
        Me.SpeakBtn.RightToLeft = System.Windows.Forms.RightToLeft.No
        Me.SpeakBtn.Size = New System.Drawing.Size(96, 32)
        Me.SpeakBtn.TabIndex = 1
        Me.SpeakBtn.Text = "Speak"
        '
        'VolumeSldr
        '
        Me.VolumeSldr.Location = New System.Drawing.Point(192, 376)
        Me.VolumeSldr.Name = "VolumeSldr"
        Me.VolumeSldr.OcxState =
CType(resources.GetObject("VolumeSldr.OcxState"),
System.Windows.Forms.AxHost.State)
        Me.VolumeSldr.Size = New System.Drawing.Size(220, 16)
        Me.VolumeSldr.TabIndex = 13
        '
        'Label1
        '
        Me.Label1.BackColor = System.Drawing.Color.Transparent
        Me.Label1.Cursor = System.Windows.Forms.Cursors.Default
```

```
    Me.Label1.Font = New System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
    Me.Label1.ForeColor = System.Drawing.SystemColors.ControlText
    Me.Label1.Location = New System.Drawing.Point(128, 400)
    Me.Label1.Name = "Label1"
    Me.Label1.RightToLeft = System.Windows.Forms.RightToLeft.No
    Me.Label1.Size = New System.Drawing.Size(48, 25)
    Me.Label1.TabIndex = 16
    Me.Label1.Text = "Audio Output"
    '
    'Label5
    '
    Me.Label5.BackColor = System.Drawing.Color.Transparent
    Me.Label5.Cursor = System.Windows.Forms.Cursors.Default
    Me.Label5.Font = New System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
    Me.Label5.ForeColor = System.Drawing.SystemColors.ControlText
    Me.Label5.Location = New System.Drawing.Point(128, 312)
    Me.Label5.Name = "Label5"
    Me.Label5.RightToLeft = System.Windows.Forms.RightToLeft.No
    Me.Label5.Size = New System.Drawing.Size(48, 17)
    Me.Label5.TabIndex = 14
    Me.Label5.Text = "Format"
    '
    'Label4
    '
    Me.Label4.BackColor = System.Drawing.Color.Transparent
    Me.Label4.Cursor = System.Windows.Forms.Cursors.Default
    Me.Label4.Font = New System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
    Me.Label4.ForeColor = System.Drawing.SystemColors.ControlText
    Me.Label4.Location = New System.Drawing.Point(128, 376)
    Me.Label4.Name = "Label4"
    Me.Label4.RightToLeft = System.Windows.Forms.RightToLeft.No
    Me.Label4.Size = New System.Drawing.Size(49, 17)
    Me.Label4.TabIndex = 12
    Me.Label4.Text = "Volume"
```

```
        '
        'Label3
        '
        Me.Label3.BackColor = System.Drawing.Color.Transparent
        Me.Label3.Cursor = System.Windows.Forms.Cursors.Default
        Me.Label3.Font = New System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
        Me.Label3.ForeColor = System.Drawing.SystemColors.ControlText
        Me.Label3.Location = New System.Drawing.Point(128, 344)
        Me.Label3.Name = "Label3"
        Me.Label3.RightToLeft = System.Windows.Forms.RightToLeft.No
        Me.Label3.Size = New System.Drawing.Size(49, 17)
        Me.Label3.TabIndex = 10
        Me.Label3.Text = "Rate"
        '
        'Label2
        '
        Me.Label2.BackColor = System.Drawing.Color.Transparent
        Me.Label2.Cursor = System.Windows.Forms.Cursors.Default
        Me.Label2.Font = New System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
        Me.Label2.ForeColor = System.Drawing.SystemColors.ControlText
        Me.Label2.Location = New System.Drawing.Point(128, 272)
        Me.Label2.Name = "Label2"
        Me.Label2.RightToLeft = System.Windows.Forms.RightToLeft.No
        Me.Label2.Size = New System.Drawing.Size(40, 17)
        Me.Label2.TabIndex = 8
        Me.Label2.Text = "Voice"
        '
        'MainMenu1
        '
        Me.MainMenu1.MenuItems.AddRange(New
System.Windows.Forms.MenuItem() {Me.menuFile, Me.menuHelp})
        '
        'menuFile
        '
        Me.menuFile.Index = 0
        Me.menuFile.MenuItems.AddRange(New
System.Windows.Forms.MenuItem() {Me.menuFileSpeakWave,
Me.menuFileSaveToWave, Me.menuSep, Me.menuFileExit})
        Me.menuFile.Text = "File"
        '
        'menuFileSpeakWave
        '
        Me.menuFileSpeakWave.Index = 0
        Me.menuFileSpeakWave.Shortcut =
System.Windows.Forms.Shortcut.CtrlW
        Me.menuFileSpeakWave.Text = "Speak Wave File"
        '
        'menuFileSaveToWave
        '
        Me.menuFileSaveToWave.Index = 1
        Me.menuFileSaveToWave.Shortcut =
System.Windows.Forms.Shortcut.CtrlS
        Me.menuFileSaveToWave.Text = "Save To Wave File"
        '
        'menuSep
        '
        Me.menuSep.Index = 2
        Me.menuSep.Text = "-"
        '
        'menuFileExit
        '
        Me.menuFileExit.Index = 3
        Me.menuFileExit.Shortcut = System.Windows.Forms.Shortcut.CtrlQ
        Me.menuFileExit.Text = "Quit"
        '
        'menuHelp
        '
        Me.menuHelp.Index = 1
        Me.menuHelp.MenuItems.AddRange(New
System.Windows.Forms.MenuItem() {Me.menuAbout})
        Me.menuHelp.Text = "Help"
        '
        'menuAbout
        '
        Me.menuAbout.Index = 0
        Me.menuAbout.Text = "About"
        '
        'AnimateFace
        '
        Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
        Me.BackColor = System.Drawing.SystemColors.Window
        Me.ClientSize = New System.Drawing.Size(730, 559)
        Me.Controls.Add(Me.VisemePicture)
        Me.Controls.Add(Me.StopBtn)
        Me.Controls.Add(Me.Frame1)
        Me.Controls.Add(Me.AudioOutputCB)
        Me.Controls.Add(Me.MainTxtBox)
        Me.Controls.Add(Me.DebugTxtBox)
        Me.Controls.Add(Me.ResetBtn)
        Me.Controls.Add(Me.MouthImgList)
        Me.Controls.Add(Me.ComDlg)
        Me.Controls.Add(Me.FormatCB)
        Me.Controls.Add(Me.RateSldr)
        Me.Controls.Add(Me.VoiceCB)
        Me.Controls.Add(Me.PauseBtn)
        Me.Controls.Add(Me.SpeakBtn)
        Me.Controls.Add(Me.VolumeSldr)
        Me.Controls.Add(Me.Label1)
        Me.Controls.Add(Me.Label5)
        Me.Controls.Add(Me.Label4)
        Me.Controls.Add(Me.Label3)
        Me.Controls.Add(Me.Label2)
        Me.Cursor = System.Windows.Forms.Cursors.Default
        Me.Font = New System.Drawing.Font("Arial", 8.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
        Me.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedSingle
        Me.Icon = CType(resources.GetObject("$this.Icon"),
System.Drawing.Icon)
        Me.Location = New System.Drawing.Point(223, 243)
        Me.MaximizeBox = False
        Me.Menu = Me.MainMenu1
        Me.Name = "AnimateFace"
        Me.RightToLeft = System.Windows.Forms.RightToLeft.No
        Me.StartPosition =
System.Windows.Forms.FormStartPosition.CenterScreen
        Me.Text = "AnimateFace"
        Me.WindowState =
System.Windows.Forms.FormWindowState.Maximized
        Me.Frame1.ResumeLayout(False)
        CType(Me.MouthImgList,
System.ComponentModel.ISupportInitialize).EndInit()
        CType(Me.ComDlg,
System.ComponentModel.ISupportInitialize).EndInit()
        CType(Me.RateSldr,
System.ComponentModel.ISupportInitialize).EndInit()
        CType(Me.VolumeSldr,
System.ComponentModel.ISupportInitialize).EndInit()
        Me.ResumeLayout(False)

    End Sub
#End Region
#Region "InterOp Services "
    Private Shared m_vb6FormDefInstance As AnimateFace
    Private Shared m_InitializingDefInstance As Boolean
    Public Shared Property DefInstance() As AnimateFace
        Get
            If m_vb6FormDefInstance Is Nothing OrElse
m_vb6FormDefInstance.IsDisposed Then
                m_InitializingDefInstance = True
                m_vb6FormDefInstance = New AnimateFace
                m_InitializingDefInstance = False
            End If
            DefInstance = m_vb6FormDefInstance
        End Get
        Set(ByVal Value As AnimateFace)
            m_vb6FormDefInstance = Value
        End Set
    End Property
#End Region


    ' First, declare the main SAPI object we are using in this sample. It is
    ' created inside Form_Load and released inside Form_Unload.
```

```
   Dim WithEvents Voice As SpeechLib.SpVoice

   ' Speak flags is a combination of bit flags. These individual bits
correspond
   ' to check boxes on the UI. So m_speakFlags should always be kept in
sync
   ' with the state of those check boxes.
   Dim m_speakFlags As SpeechLib.SpeechVoiceSpeakFlags

   ' This is the default format we will use.
   Const DefaultFmt As String = "SAFT22kHz16BitMono"

   ' We will disable the output combo box and show this if there's no audio
output.
   Const NoAudioOutput As String = "No audio ouput object available"

   ' We will enable/disable menu items and buttons based on current state
   ' m_speaking indicates whether a speak task is in progress
   ' m_paused indicates whether Voice.Pause is called
   Private m_bSpeaking As Boolean
   Private m_bPaused As Boolean


   Private Sub FacialAnimation_Load(ByVal eventSender As
System.Object, ByVal eventArgs As System.EventArgs) Handles
MyBase.Load
       On Error GoTo ErrHandler

       ' Creates the voice object first
       Voice = New SpeechLib.SpVoice

       ' Load the voices combo box
       Dim Token As SpeechLib.ISpeechObjectToken

       For Each Token In Voice.GetVoices
           VoiceCB.Items.Add((Token.GetDescription()))
       Next Token
       VoiceCB.SelectedIndex = 1
       VoiceCB.Enabled = False
       If VoiceCB.Items.IsReadOnly = True Then
```

```
    End If

    'load the format combo box
    AddItemToFmtCB()

    ' set rate and volume to the same as the Voice
    RateSldr.Value = Voice.Rate
    VolumeSldr.Value = Voice.Volume

    'set the default format
    FormatCB.Text = DefaultFmt

    ' Load the audio output combo box
    If Voice.GetAudioOutputs.Count > 0 Then
        For Each Token In Voice.GetAudioOutputs
            AudioOutputCB.Items.Add((Token.GetDescription))
        Next Token
    Else
        AudioOutputCB.Items.Add(NoAudioOutput)
        AudioOutputCB.Enabled = False
    End If
    AudioOutputCB.SelectedIndex = 0

    'load image list
    LoadMouthImages()

    MouthImgList.MaskColor = System.Drawing.Color.Magenta
    MouthImgList.BackColor =
System.Drawing.ColorTranslator.FromOle(GetSysColor(COLOR_3DFACE
))
    VisemePicture.Image = MouthImgList.Overlay("MICFULL",
"MICFULL")

    ' init speak flags and sync flag check boxes
    m_speakFlags = SpeechLib.SpeechVoiceSpeakFlags.SVSFlagsAsync
Or SpeechLib.SpeechVoiceSpeakFlags.SVSFPurgeBeforeSpeak Or
SpeechLib.SpeechVoiceSpeakFlags.SVSFIsXML
    chkSpFlagAync.CheckState =
System.Windows.Forms.CheckState.Checked
```

```
        SetSpeakingState(False, False)
        Exit Sub

ErrHandler:
        MsgBox("Error in initialization: " & vbCrLf & vbCrLf &
Err.Description & vbCrLf & vbCrLf & "Shutting down.",
MsgBoxStyle.OKOnly, "TTSApp")
        Voice = Nothing
        End
    End Sub

    Private Sub FacialAnimation_Closed(ByVal eventSender As
System.Object, ByVal eventArgs As System.EventArgs) Handles
MyBase.Closed
        Voice = Nothing
    End Sub

    Private Sub AudioOutputCB_SelectedIndexChanged(ByVal eventSender
As System.Object, ByVal eventArgs As System.EventArgs) Handles
AudioOutputCB.SelectedIndexChanged
        On Error GoTo ErrHandler

        ' change the output to the selected one
        Voice.AudioOutput =
Voice.GetAudioOutputs().Item(AudioOutputCB.SelectedIndex)

        ' changing output may have also changed the format, so call function
        ' FormatCB_Click to make sure we are using the format as selected
        FormatCB_SelectedIndexChanged(FormatCB, New
System.EventArgs)
        Exit Sub

ErrHandler:
        AddDebugInfo("Set audio output error: ", Err.Description)
    End Sub

    Private Sub FormatCB_SelectedIndexChanged(ByVal eventSender As
System.Object, ByVal eventArgs As System.EventArgs) Handles
FormatCB.SelectedIndexChanged
        On Error GoTo ErrHandler
```

```
   Public Sub menuFileSaveToWave_Popup(ByVal eventSender As
System.Object, ByVal eventArgs As System.EventArgs) Handles
menuFileSaveToWave.Popup
      menuFileSaveToWave_Click(eventSender, eventArgs)
   End Sub
   Public Sub menuFileSaveToWave_Click(ByVal eventSender As
System.Object, ByVal eventArgs As System.EventArgs) Handles
menuFileSaveToWave.Click
      ' Set CancelError is True
      ComDlg.CancelError = True
      On Error GoTo ErrHandler

      ' Set flags
      ComDlg.Flags =
MSComDlg.FileOpenConstants.cdlOFNOverwritePrompt Or
MSComDlg.FileOpenConstants.cdlOFNPathMustExist Or
MSComDlg.FileOpenConstants.cdlOFNNoReadOnlyReturn
      ' Set Dialog title
      ComDlg.DialogTitle = "Save to a Wave File"
      ' Set filters
      ComDlg.Filter = "All Files (*.*)|*.*|Wave Files " & "(*.wav)|*.wav"
      ' Specify default filter
      ComDlg.FilterIndex = 2
      ' Display the Open dialog box
      ComDlg.ShowSave()

      ' create a wave stream
      Dim cpFileStream As New SpeechLib.SpFileStream

      ' Set output format to selected format
      cpFileStream.Format.Type = VB6.GetItemData(FormatCB,
FormatCB.SelectedIndex)

      ' Open the file for write
      cpFileStream.Open(ComDlg.FileName,
SpeechLib.SpeechStreamFileMode.SSFMCreateForWrite, False)

      ' Set output stream to the file stream
      Voice.AllowAudioOutputFormatChangesOnNextSet = False
      Voice.AudioOutputStream = cpFileStream
```

```
        ' show action
        AddDebugInfo("Save to .wav file")

        ' speak the given text with given flags
        Voice.Speak(MainTxtBox.Text, m_speakFlags)

        ' wait until it's done speaking with a really really long timeout.
        ' the tiemout value is in unit of millisecond. -1 means forever.
        Voice.WaitUntilDone(-1)

        ' Since the output stream was set to the file stream, we need to
        ' set back to the selected audio output by calling AudioOutputCB_Click
        ' as if user just changed it through UI
        AudioOutputCB_SelectedIndexChanged(AudioOutputCB, New
System.EventArgs)

        ' close the file stream
        cpFileStream.Close()
        cpFileStream = Nothing

        MsgBox("WAV file successfully written!", MsgBoxStyle.OKOnly,
"File Saved")
        Exit Sub

ErrHandler:
        'User pressed the Cancel button, do not show error
        If Not (Err.Number = 32755) Then
            AddDebugInfo("Save to Wave file Error: ", Err.Description)
        End If

        If Not cpFileStream Is Nothing Then
            cpFileStream = Nothing
        End If
    End Sub

    Public Sub menuFileSpeakWave_Popup(ByVal eventSender As
System.Object, ByVal eventArgs As System.EventArgs) Handles
menuFileSpeakWave.Popup
        menuFileSpeakWave_Click(eventSender, eventArgs)
```

```
   End Sub
   Public Sub menuFileSpeakWave_Click(ByVal eventSender As
System.Object, ByVal eventArgs As System.EventArgs) Handles
menuFileSpeakWave.Click
      ' Set CancelError is True
      ComDlg.CancelError = True
      On Error GoTo ErrHandler
      ' Set flags
      ComDlg.Flags =
MSComDlg.FileOpenConstants.cdlOFNFileMustExist Or
MSComDlg.FileOpenConstants.cdlOFNPathMustExist
      ' Set Dialog title
      ComDlg.DialogTitle = "Speak a Wave File"
      ' Set filters
      ComDlg.Filter = "All Files (*.*)|*.*|Wave Files " & "(*.wav)|*.wav"
      ' Specify default filter
      ComDlg.FilterIndex = 2
      ' Display the Open dialog box
      ComDlg.ShowOpen()

      AddDebugInfo("Speak .wav file")

      ' Speak the contents of the wavefile. Notice here we are passing in the
      ' file name so the filename flag is set.
      MainTxtBox.Text = ComDlg.FileName

      SpeakBtn_Click(SpeakBtn, New System.EventArgs)

      Exit Sub

 ErrHandler:
      'User pressed the Cancel button, do not show error
      If Not (Err.Number = 32755) Then
         AddDebugInfo("Speak Wave Error: ", Err.Description)
      End If

      SetSpeakingState(False, m_bPaused)
      Exit Sub
   End Sub
   Private Sub PauseBtn_Click(ByVal eventSender As System.Object,
ByVal eventArgs As System.EventArgs) Handles PauseBtn.Click
      Select Case PauseBtn.Text
         Case "Pause"
            AddDebugInfo("Pause")
            Voice.Pause()
            SetSpeakingState(m_bSpeaking, True)

         Case "Resume"
            AddDebugInfo("Resume")
            Voice.Resume()
            SetSpeakingState(m_bSpeaking, False)
      End Select
   End Sub

   Private Sub RateSldr_Scroll(ByVal eventSender As System.Object,
ByVal eventArgs As System.EventArgs) Handles RateSldr.Scroll
      Voice.Rate = RateSldr.Value
   End Sub

   Private Sub ResetBtn_Click(ByVal eventSender As System.Object,
ByVal eventArgs As System.EventArgs) Handles ResetBtn.Click
      'set output to default
      AudioOutputCB.SelectedIndex = 0
      Voice.AudioOutput = Nothing

      'use default voice
      VoiceCB.SelectedIndex = 0

      'Format to default
      FormatCB.Text = DefaultFmt

      'reset main text field
      MainTxtBox.Text = "Enter text you wish spoken here."

      'reset volume and rate
      VolumeSldr.Value = 100
      VolumeSldr_Scroll(VolumeSldr, New System.EventArgs)

      RateSldr.Value = 0
```

```
        RateSldr_Scroll(RateSldr, New System.EventArgs)

        ' reset speak flags
        m_speakFlags = SpeechLib.SpeechVoiceSpeakFlags.SVSFlagsAsync
Or SpeechLib.SpeechVoiceSpeakFlags.SVSFPurgeBeforeSpeak Or
SpeechLib.SpeechVoiceSpeakFlags.SVSFIsXML
        chkSpFlagAync.CheckState =
System.Windows.Forms.CheckState.Checked

        chkSpFlagNLPSpeakPunc.CheckState =
System.Windows.Forms.CheckState.Unchecked

        'reset DebugTxtbox text
        DebugTxtBox.Text = CStr(Nothing)

        'reset skip text box
        NumericUpDown1.Text = "0"
        VisemePicture.Image = MouthImgList.Overlay("MICFULL",
"MICFULL")

        ' if it's paused, call Resume to reset state
        If m_bPaused Then Voice.Resume()

        SetSpeakingState(False, False)
    End Sub

    Private Sub SkipBtn_Click(ByVal eventSender As System.Object, ByVal
eventArgs As System.EventArgs)
        On Error GoTo ErrHandler
        Dim SkipType As String
        Dim SkipNum As Short

        AddDebugInfo("Skip")

        ' skip by the number specified
        SkipNum = CShort(NumericUpDown1.Text)
        SkipType = "Sentence"

        Voice.Skip(SkipType, SkipNum)
        Exit Sub
```

```
ErrHandler:
      'MsgBox Err.Description & ":" & Err.Number, vbOKOnly, "Skip
Error"
      AddDebugInfo("Skip Error: ", Err.Description)
      Exit Sub
   End Sub

   Private Sub SpeakBtn_Click(ByVal eventSender As System.Object,
ByVal eventArgs As System.EventArgs) Handles SpeakBtn.Click
      On Error GoTo ErrHandler
      AddDebugInfo(("Speak"))

      ' exit if there's nothing to speak
      If MainTxtBox.Text = "" Then
         Exit Sub
      End If

      ' If it's paused and some text still remains to be spoken, Speak button
      ' acts the same as Resume button. However a programmer can choose
to
      ' speak from the beginning again or any other behavior.
      ' In other cases, we speak the text with given flags.
      If Not (m_bPaused And m_bSpeaking) Then
         ' just speak the text with the given flags
         Voice.Speak(MainTxtBox.Text, m_speakFlags)
      End If

      ' Resume if Voice is paused
      If m_bPaused Then Voice.Resume()

      ' set the state of menu items and buttons
      SetSpeakingState(True, False)
      Exit Sub

 ErrHandler:
      AddDebugInfo("Speak Error: ", Err.Description)
      SetSpeakingState(False, m_bPaused)
   End Sub
   Private Sub StopBtn_Click(ByVal eventSender As System.Object, ByVal
eventArgs As System.EventArgs) Handles StopBtn.Click
        On Error GoTo ErrHandler
        AddDebugInfo(("Stop"))

        ' when string to speak is NULL and dwFlags is set to
SPF_PURGEBEFORESPEAK
        ' it indicates to SAPI that any remaining data to be synthesized should
        ' be discarded.
        Voice.Speak(vbNullString,
SpeechLib.SpeechVoiceSpeakFlags.SVSFPurgeBeforeSpeak)
        If m_bPaused Then Voice.Resume()

        SetSpeakingState(False, False)
        Exit Sub

ErrHandler:
        AddDebugInfo("Speak Error: ", Err.Description)
   End Sub

   Private Sub Voice_AudioLevel(ByVal StreamNumber As Integer, ByVal
StreamPosition As Object, ByVal AudioLevel As Integer) Handles
Voice.AudioLevel
        ShowEvent("AudioLevel", "StreamNumber=" & StreamNumber,
"StreamPosition=" & StreamPosition, "AudioLevel=" & AudioLevel)
   End Sub

   Private Sub Voice_Bookmark(ByVal StreamNumber As Integer, ByVal
StreamPosition As Object, ByVal Bookmark As String, ByVal BookmarkId
As Integer) Handles Voice.Bookmark
        ShowEvent("BookMark", "StreamNumber=" & StreamNumber,
"StreamPosition=" & StreamPosition, "Bookmark=" & Bookmark,
"BookmarkId=" & BookmarkId)
   End Sub

   Private Sub Voice_EndStream(ByVal StreamNum As Integer, ByVal
StreamPos As Object) Handles Voice.EndStream
        ShowEvent("EndStream", "StreamNum=" & StreamNum,
"StreamPos=" & StreamPos)
```

```
        ' select all text to indicate that we are done
        HighLightSpokenWords(0, Len(MainTxtBox.Text))

        ' reset the mouth
        VisemePicture.Image = MouthImgList.Overlay("MICFULL",
"MICFULL")

        ' reset the state of buttons, checkboxes and menu items
        SetSpeakingState(False, m_bPaused)
    End Sub

    Private Sub Voice_EnginePrivate(ByVal StreamNumber As Integer,
ByVal StreamPosition As Integer, ByVal lParam As Object) Handles
Voice.EnginePrivate
        ShowEvent("EnginePrivate", "StreamNumber=" & StreamNumber,
"StreamPosition=" & StreamPosition, "lParam=" & lParam)
    End Sub

    Private Sub Voice_Phoneme(ByVal StreamNumber As Integer, ByVal
StreamPosition As Object, ByVal Duration As Integer, ByVal NextPhoneId
As Short, ByVal Feature As SpeechLib.SpeechVisemeFeature, ByVal
CurrentPhoneId As Short) Handles Voice.Phoneme
        ShowEvent("Phoneme", "StreamNumber=" & StreamNumber,
"StreamPosition=" & StreamPosition, "NextPhoneId=" & NextPhoneId,
"Feature=" & Feature, "CurrentPhoneId=" & CurrentPhoneId)
    End Sub

    Private Sub Voice_Sentence(ByVal StreamNum As Integer, ByVal
StreamPos As Object, ByVal Pos As Integer, ByVal Length As Integer)
Handles Voice.Sentence
        ShowEvent("Sentence", "StreamNum=" & StreamNum, "StreamPos="
& StreamPos, "Pos=" & Pos, "Length=" & Length)
    End Sub

    Private Sub Voice_StartStream(ByVal StreamNum As Integer, ByVal
StreamPos As Object) Handles Voice.StartStream
        ShowEvent("StartStream", "StreamNum=" & StreamNum,
"StreamPos=" & StreamPos)

        ' reset the state of buttons, checkboxes and menu items
        SetSpeakingState(True, m_bPaused)
    End Sub

    Private Sub Voice_Viseme(ByVal StreamNum As Integer, ByVal
StreamPos As Object, ByVal Duration As Integer, ByVal VisemeType As
SpeechLib.SpeechVisemeType, ByVal Feature As
SpeechLib.SpeechVisemeFeature, ByVal VisemeId As
SpeechLib.SpeechVisemeType) Handles Voice.Viseme

        ShowEvent("Viseme", "StreamNum=" & StreamNum, "StreamPos=" &
StreamPos, "Duration=" & Duration, "VisemeType=" & VisemeType,
"Feature=" & Feature, "VisemeId=" & VisemeId)

        ' Here we are going to show different mouth positions according to the
viseme.
        ' The picture we show doesn't necessarily match the real mouth
position.
        ' Just trying to make it more interesting.
        If VisemeId = 0 Then
            VisemeId = VisemeId + 2
        End If
        VisemePicture.Image = MouthImgList.Overlay("MICFULL",
VisemeId)
        If (VisemeId Mod 6 = 2) Then
            VisemePicture.Image = MouthImgList.Overlay("MICFULL",
"MICEYECLOSED")
        Else
            If (VisemeId Mod 6 = 5) Then
                VisemePicture.Image = MouthImgList.Overlay("MICFULL",
"MICEYENARROW")
            End If
        End If
    End Sub

    Private Sub Voice_VoiceChange(ByVal StreamNum As Integer, ByVal
StreamPos As Object, ByVal Token As SpeechLib.SpObjectToken) Handles
Voice.VoiceChange

        ShowEvent("VoiceChange", "StreamNum=" & StreamNum,
"StreamPos=" & StreamPos, "Token=" & Token.GetDescription)
```

```
        ' Let's sync up the combo box with the new value
        Dim i As Integer
        For i = 0 To VoiceCB.Items.Count - 1
            If VB6.GetItemString(VoiceCB, i) = Token.GetDescription() Then
                VoiceCB.SelectedIndex = i
                Exit For
            End If
        Next
    End Sub

    Private Sub Voice_Word(ByVal StreamNum As Integer, ByVal
StreamPos As Object, ByVal Pos As Integer, ByVal Length As Integer)
Handles Voice.Word

        ShowEvent("Word", "StreamNum=" & StreamNum, "StreamPos=" &
StreamPos, "Pos=" & Pos, "Length=" & Length)

        System.Diagnostics.Debug.WriteLine(VB6.TabLayout(Pos, Length,
MainTxtBox.SelectionStart, MainTxtBox.SelectionLength))

        ' Select the word that's currently being spoken.
        HighLightSpokenWords(Pos, Length)
    End Sub

    Private Sub VoiceCB_SelectedIndexChanged(ByVal eventSender As
System.Object, ByVal eventArgs As System.EventArgs) Handles
VoiceCB.SelectedIndexChanged
        ' change the voice to the selected one
        Voice.Voice = Voice.GetVoices().Item(VoiceCB.SelectedIndex)
    End Sub

    Private Sub VolumeSldr_Scroll(ByVal eventSender As System.Object,
ByVal eventArgs As System.EventArgs) Handles VolumeSldr.Scroll
        Voice.Volume = VolumeSldr.Value
    End Sub

    ' The following functions are simply to sync up the speak flags.
    ' When the check box is checked, the corresponding bit is set in the flags.
```

```
   Private Sub chkSpFlagAync_CheckStateChanged(ByVal eventSender As
System.Object, ByVal eventArgs As System.EventArgs) Handles
chkSpFlagAync.CheckStateChanged
      m_speakFlags = SetOrClearFlag(chkSpFlagAync.CheckState,
m_speakFlags, SpeechLib.SpeechVoiceSpeakFlags.SVSFlagsAsync)
   End Sub



   Private Sub chkSpFlagIsXML_CheckStateChanged(ByVal eventSender
As System.Object, ByVal eventArgs As System.EventArgs)
      ' Note: special case here. There are two flags,SVSFIsXML and
SVSFIsNotXML.
      ' When neither is set, SAPI will guess by peeking at beginning
characters.
      ' In this sample, we explicitly set one of them.


      ' clear SVSFIsNotXML bit and set SVSFIsXML bit
      m_speakFlags = m_speakFlags And Not
SpeechLib.SpeechVoiceSpeakFlags.SVSFIsNotXML
      m_speakFlags = m_speakFlags Or
SpeechLib.SpeechVoiceSpeakFlags.SVSFIsXML

   End Sub

   Private Sub chkSpFlagNLPSpeakPunc_CheckStateChanged(ByVal
eventSender As System.Object, ByVal eventArgs As System.EventArgs)
Handles chkSpFlagNLPSpeakPunc.CheckStateChanged
      m_speakFlags =
SetOrClearFlag(chkSpFlagNLPSpeakPunc.CheckState, m_speakFlags,
SpeechLib.SpeechVoiceSpeakFlags.SVSFNLPSpeakPunc)
   End Sub
   Private Sub AddFmts(ByRef name_Renamed As String, ByVal fmt As
SpeechLib.SpeechAudioFormatType)
      Dim index As String
      ' get the count of existing list so that we are adding to the bottom of the
list
      index = CStr(FormatCB.Items.Count)
```

```
        ' add the name to the list box and associate the format type with the
item
        FormatCB.Items.Insert(index, name_Renamed)
        VB6.SetItemData(FormatCB, index, fmt)
    End Sub

    Private Sub AddItemToFmtCB()
        AddFmts("SAFT8kHz8BitMono",
SpeechLib.SpeechAudioFormatType.SAFT8kHz16BitMono)
        AddFmts("SAFT8kHz8BitStereo",
SpeechLib.SpeechAudioFormatType.SAFT8kHz8BitStereo)
        AddFmts("SAFT8kHz16BitMono",
SpeechLib.SpeechAudioFormatType.SAFT8kHz16BitMono)
        AddFmts("SAFT8kHz16BitStereo",
SpeechLib.SpeechAudioFormatType.SAFT8kHz16BitStereo)

        AddFmts("SAFT11kHz8BitMono",
SpeechLib.SpeechAudioFormatType.SAFT11kHz8BitMono)
        AddFmts("SAFT11kHz8BitStereo",
SpeechLib.SpeechAudioFormatType.SAFT11kHz8BitStereo)
        AddFmts("SAFT11kHz16BitMono",
SpeechLib.SpeechAudioFormatType.SAFT11kHz16BitMono)
        AddFmts("SAFT11kHz16BitStereo",
SpeechLib.SpeechAudioFormatType.SAFT11kHz16BitStereo)

        AddFmts("SAFT12kHz8BitMono",
SpeechLib.SpeechAudioFormatType.SAFT12kHz8BitMono)
        AddFmts("SAFT12kHz8BitStereo",
SpeechLib.SpeechAudioFormatType.SAFT12kHz8BitStereo)
        AddFmts("SAFT12kHz16BitMono",
SpeechLib.SpeechAudioFormatType.SAFT12kHz16BitMono)
        AddFmts("SAFT12kHz16BitStereo",
SpeechLib.SpeechAudioFormatType.SAFT12kHz16BitStereo)

        AddFmts("SAFT16kHz8BitMono",
SpeechLib.SpeechAudioFormatType.SAFT16kHz8BitMono)
        AddFmts("SAFT16kHz8BitStereo",
SpeechLib.SpeechAudioFormatType.SAFT16kHz8BitStereo)
        AddFmts("SAFT16kHz16BitMono",
SpeechLib.SpeechAudioFormatType.SAFT16kHz16BitMono)
```

```
      AddFmts("SAFT16kHz16BitStereo",
 SpeechLib.SpeechAudioFormatType.SAFT16kHz16BitStereo)

      AddFmts("SAFT22kHz8BitMono",
 SpeechLib.SpeechAudioFormatType.SAFT22kHz8BitMono)
      AddFmts("SAFT22kHz8BitStereo",
 SpeechLib.SpeechAudioFormatType.SAFT22kHz8BitStereo)
      AddFmts("SAFT22kHz16BitMono",
 SpeechLib.SpeechAudioFormatType.SAFT22kHz16BitMono)
      AddFmts("SAFT22kHz16BitStereo",
 SpeechLib.SpeechAudioFormatType.SAFT22kHz16BitStereo)

      AddFmts("SAFT24kHz8BitMono",
 SpeechLib.SpeechAudioFormatType.SAFT24kHz8BitMono)
      AddFmts("SAFT24kHz8BitStereo",
 SpeechLib.SpeechAudioFormatType.SAFT24kHz8BitStereo)
      AddFmts("SAFT24kHz16BitMono",
 SpeechLib.SpeechAudioFormatType.SAFT24kHz16BitMono)
      AddFmts("SAFT24kHz16BitStereo",
 SpeechLib.SpeechAudioFormatType.SAFT24kHz16BitStereo)

      AddFmts("SAFT32kHz8BitMono",
 SpeechLib.SpeechAudioFormatType.SAFT32kHz8BitMono)
      AddFmts("SAFT32kHz8BitStereo",
 SpeechLib.SpeechAudioFormatType.SAFT32kHz8BitStereo)
      AddFmts("SAFT32kHz16BitMono",
 SpeechLib.SpeechAudioFormatType.SAFT32kHz16BitMono)
      AddFmts("SAFT32kHz16BitStereo",
 SpeechLib.SpeechAudioFormatType.SAFT32kHz16BitStereo)

      AddFmts("SAFT44kHz8BitMono",
 SpeechLib.SpeechAudioFormatType.SAFT44kHz8BitMono)
      AddFmts("SAFT44kHz8BitStereo",
 SpeechLib.SpeechAudioFormatType.SAFT44kHz8BitStereo)
      AddFmts("SAFT44kHz16BitMono",
 SpeechLib.SpeechAudioFormatType.SAFT44kHz16BitMono)
      AddFmts("SAFT44kHz16BitStereo",
 SpeechLib.SpeechAudioFormatType.SAFT44kHz16BitStereo)
```

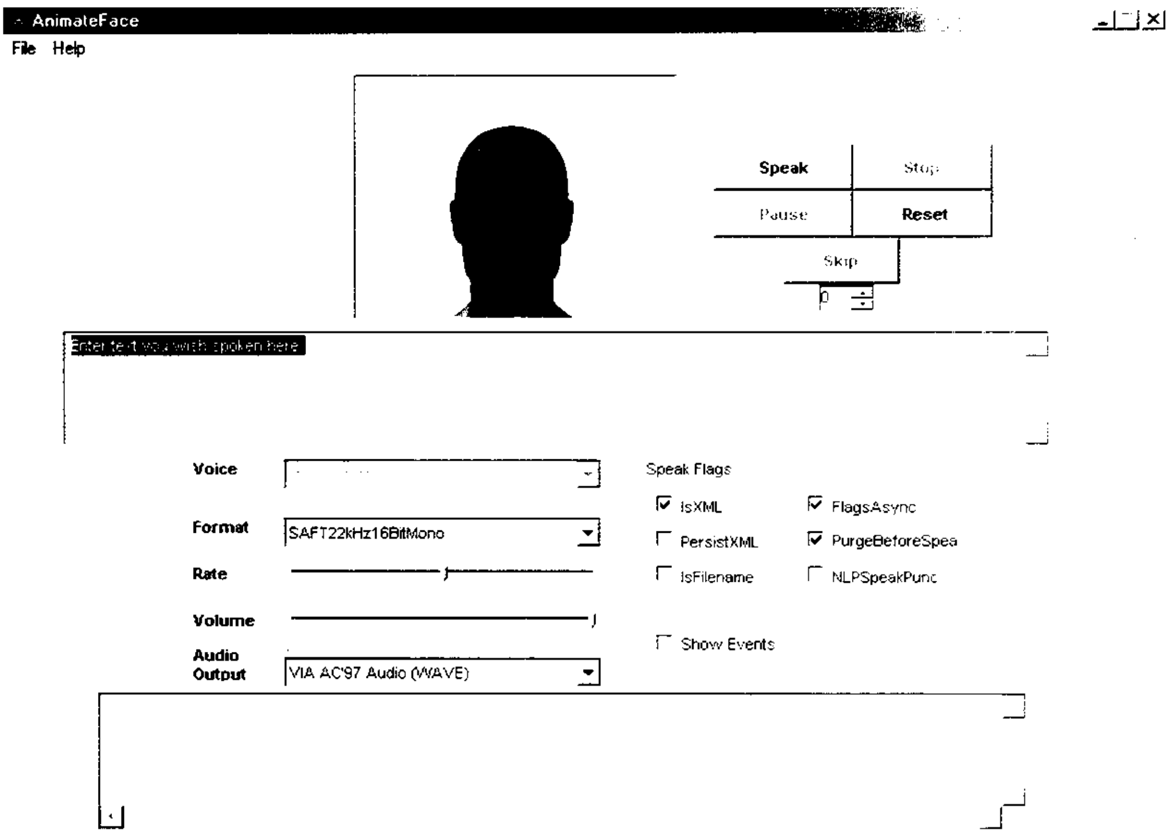
```
      AddFmts("SAFT48kHz8BitMono",
SpeechLib.SpeechAudioFormatType.SAFT48kHz8BitMono)
      AddFmts("SAFT48kHz8BitStereo",
SpeechLib.SpeechAudioFormatType.SAFT48kHz8BitStereo)
      AddFmts("SAFT48kHz16BitMono",
SpeechLib.SpeechAudioFormatType.SAFT48kHz16BitMono)
      AddFmts("SAFT48kHz16BitStereo",
SpeechLib.SpeechAudioFormatType.SAFT48kHz16BitStereo)
   End Sub
   Private Sub LoadMouthImages()
      On Error GoTo ErrHandler

      MouthImgList.ListImages.Add(1, "MICFULL",
VB6.ImageToIPictureDisp(VB6.LoadResPicture("MICFULL",
VB6.LoadResConstants.ResBitmap)))
      MouthImgList.ListImages.Add(2, ,
VB6.ImageToIPictureDisp(VB6.LoadResPicture("MIC11",
VB6.LoadResConstants.ResBitmap)))
      MouthImgList.ListImages.Add(3, ,
VB6.ImageToIPictureDisp(VB6.LoadResPicture("MIC11",
VB6.LoadResConstants.ResBitmap)))
      MouthImgList.ListImages.Add(4, ,
VB6.ImageToIPictureDisp(VB6.LoadResPicture("MIC11",
VB6.LoadResConstants.ResBitmap)))
      MouthImgList.ListImages.Add(5, ,
VB6.ImageToIPictureDisp(VB6.LoadResPicture("MIC10",
VB6.LoadResConstants.ResBitmap)))
      MouthImgList.ListImages.Add(6, ,
VB6.ImageToIPictureDisp(VB6.LoadResPicture("MIC11",
VB6.LoadResConstants.ResBitmap)))
      MouthImgList.ListImages.Add(7, ,
VB6.ImageToIPictureDisp(VB6.LoadResPicture("MIC9",
VB6.LoadResConstants.ResBitmap)))
      MouthImgList.ListImages.Add(8, ,
VB6.ImageToIPictureDisp(VB6.LoadResPicture("MIC2",
VB6.LoadResConstants.ResBitmap)))
      MouthImgList.ListImages.Add(9, ,
VB6.ImageToIPictureDisp(VB6.LoadResPicture("MIC13",
VB6.LoadResConstants.ResBitmap)))
```

```
    MouthImgList.ListImages.Add(10, ,
VB6.ImageToIPictureDisp(VB6.LoadResPicture("MIC9",
VB6.LoadResConstants.ResBitmap)))
    MouthImgList.ListImages.Add(11, ,
VB6.ImageToIPictureDisp(VB6.LoadResPicture("MIC12",
VB6.LoadResConstants.ResBitmap)))
    MouthImgList.ListImages.Add(12, ,
VB6.ImageToIPictureDisp(VB6.LoadResPicture("MIC11",
VB6.LoadResConstants.ResBitmap)))
    MouthImgList.ListImages.Add(13, ,
VB6.ImageToIPictureDisp(VB6.LoadResPicture("MIC9",
VB6.LoadResConstants.ResBitmap)))
    MouthImgList.ListImages.Add(14, ,
VB6.ImageToIPictureDisp(VB6.LoadResPicture("MIC3",
VB6.LoadResConstants.ResBitmap)))
    MouthImgList.ListImages.Add(15, ,
VB6.ImageToIPictureDisp(VB6.LoadResPicture("MIC6",
VB6.LoadResConstants.ResBitmap)))
    MouthImgList.ListImages.Add(16, ,
VB6.ImageToIPictureDisp(VB6.LoadResPicture("MIC7",
VB6.LoadResConstants.ResBitmap)))
    MouthImgList.ListImages.Add(17, ,
VB6.ImageToIPictureDisp(VB6.LoadResPicture("MIC8",
VB6.LoadResConstants.ResBitmap)))
    MouthImgList.ListImages.Add(18, ,
VB6.ImageToIPictureDisp(VB6.LoadResPicture("MIC5",
VB6.LoadResConstants.ResBitmap)))
    MouthImgList.ListImages.Add(19, ,
VB6.ImageToIPictureDisp(VB6.LoadResPicture("MIC4",
VB6.LoadResConstants.ResBitmap)))
    MouthImgList.ListImages.Add(20, ,
VB6.ImageToIPictureDisp(VB6.LoadResPicture("MIC7",
VB6.LoadResConstants.ResBitmap)))
    MouthImgList.ListImages.Add(21, ,
VB6.ImageToIPictureDisp(VB6.LoadResPicture("MIC9",
VB6.LoadResConstants.ResBitmap)))
    MouthImgList.ListImages.Add(22, ,
VB6.ImageToIPictureDisp(VB6.LoadResPicture("MIC11",
VB6.LoadResConstants.ResBitmap)))
```

```
      MouthImgList.ListImages.Add(23, "MICEYECLOSED",
VB6.ImageToIPictureDisp(VB6.LoadResPicture("MICEYECLOSED",
VB6.LoadResConstants.ResBitmap)))
      MouthImgList.ListImages.Add(24, "MICEYENARROW",
VB6.ImageToIPictureDisp(VB6.LoadResPicture("MICEYENARROW",
VB6.LoadResConstants.ResBitmap)))

      Exit Sub
ErrHandler:
      MsgBox(Err.Description & ":" & Err.Number, MsgBoxStyle.OKOnly,
"Load Images Error")
   End Sub

   Private Sub AddDebugInfo(ByRef DebugStr As String, Optional ByRef
Error_Renamed As String = "")
      ' This function adds debug string to the info window.

      ' First of all, let's delete a few charaters if the text box is about to
      ' overflow. In this sample we are using the default limit of charaters.
      If Len(DebugTxtBox.Text) > 64000 Then
         System.Diagnostics.Debug.WriteLine("Too much stuff in the debug
window. Remove first 10K chars")
         DebugTxtBox.SelectionStart = 0
         DebugTxtBox.SelectionLength = 10240
         DebugTxtBox.SelectedText = ""
      End If

      ' append the string to the DebugTxtBox text box and add a newline
      DebugTxtBox.SelectionStart = Len(DebugTxtBox.Text)
      DebugTxtBox.SelectedText = DebugStr & Error_Renamed & vbCrLf
   End Sub

   Private Sub ShowEvent(ByVal ParamArray strArray() As Object)
      ' we will only show the events if the ShowEvents box is checked
      Dim strText As String
      If chkShowEvents.CheckState =
System.Windows.Forms.CheckState.Checked Then
         strText = Join(strArray, ", ")
         AddDebugInfo("  Event: " & strText)
      End If
```

```
  End Sub

  Private Sub HighLightSpokenWords(ByVal Pos As Integer, ByVal
Length As Integer)
      On Error GoTo ErrHandler

      ' Only high light when the MainTxtBox is actually showing the spoken
text,
      ' instead of file name



      Exit Sub

 ErrHandler:
      AddDebugInfo("Failed to high light words. This may be caused by too
 many charaters in the main text box.")
   End Sub

   ' This following helper function will set or clear a bit (flag) in the given
   ' integer (base) according to the condition (cond). If cond is 0, the bit
   ' is cleared. Otherwise, the bit is set. The resulting integer is returned.
   Private Function SetOrClearFlag(ByVal cond As Integer, ByVal base As
 Integer, ByVal flag As Integer) As Integer

       If cond = 0 Then
          ' the condition is false, clear the flag
          SetOrClearFlag = base And Not flag
       Else
          ' the condition is false, set the flag
          SetOrClearFlag = base Or flag
       End If
    End Function

    Private Sub SetSpeakingState(ByVal bSpeaking As Boolean, ByVal
  bPaused As Boolean)
        ' change state of menu items and buttons accordingly
        menuFileSpeakWave.Enabled = Not bSpeaking
        menuFileSaveToWave.Enabled = Not bSpeaking

        SpeakBtn.Enabled = True
        StopBtn.Enabled = bSpeaking
        SkipBtn.Enabled = (bSpeaking And Not bPaused)
        PauseBtn.Enabled = bSpeaking

        If bPaused Then
           PauseBtn.Text = "Resume"
        Else
           PauseBtn.Text = "Pause"
        End If

        m_bSpeaking = bSpeaking
        m_bPaused = bPaused
     End Sub

     Public Function GetDirectory() As String

        Err.Clear()

        On Error GoTo ErrHandler

        Dim DataKey As SpeechLib.ISpeechDataKey
        Dim Category As New SpeechLib.SpObjectTokenCategory

        'Get the sdk installation location from the registry
        'The value is under
  "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Speech". The string
  name is SDKPath"

  Category.SetId(SpeechLib.SpeechStringConstants.SpeechRegistryLocalMac
  hineRoot)
        DataKey = Category.GetDataKey
        GetDirectory = DataKey.GetStringValue("SDKPath")
        GetDirectory = GetDirectory & "samples\common"
  ErrHandler:
        If Err.Number <> 0 Then
           GetDirectory = ""
        End If
     End Function
  End Class
```

11.2 SAMPLE OUTPUT



Enter text you wish spoken here

Voice

Format: SAFT22kHz16BitMono

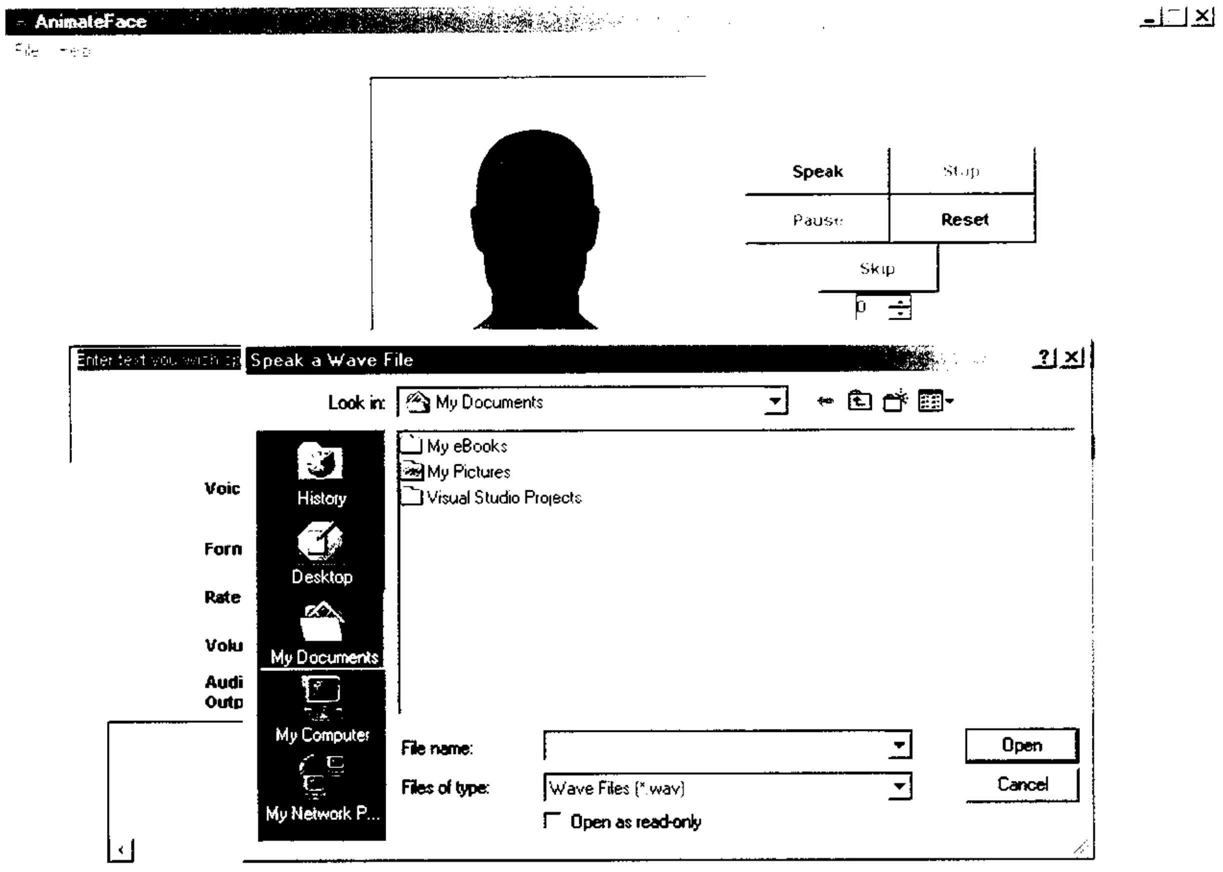
Rate

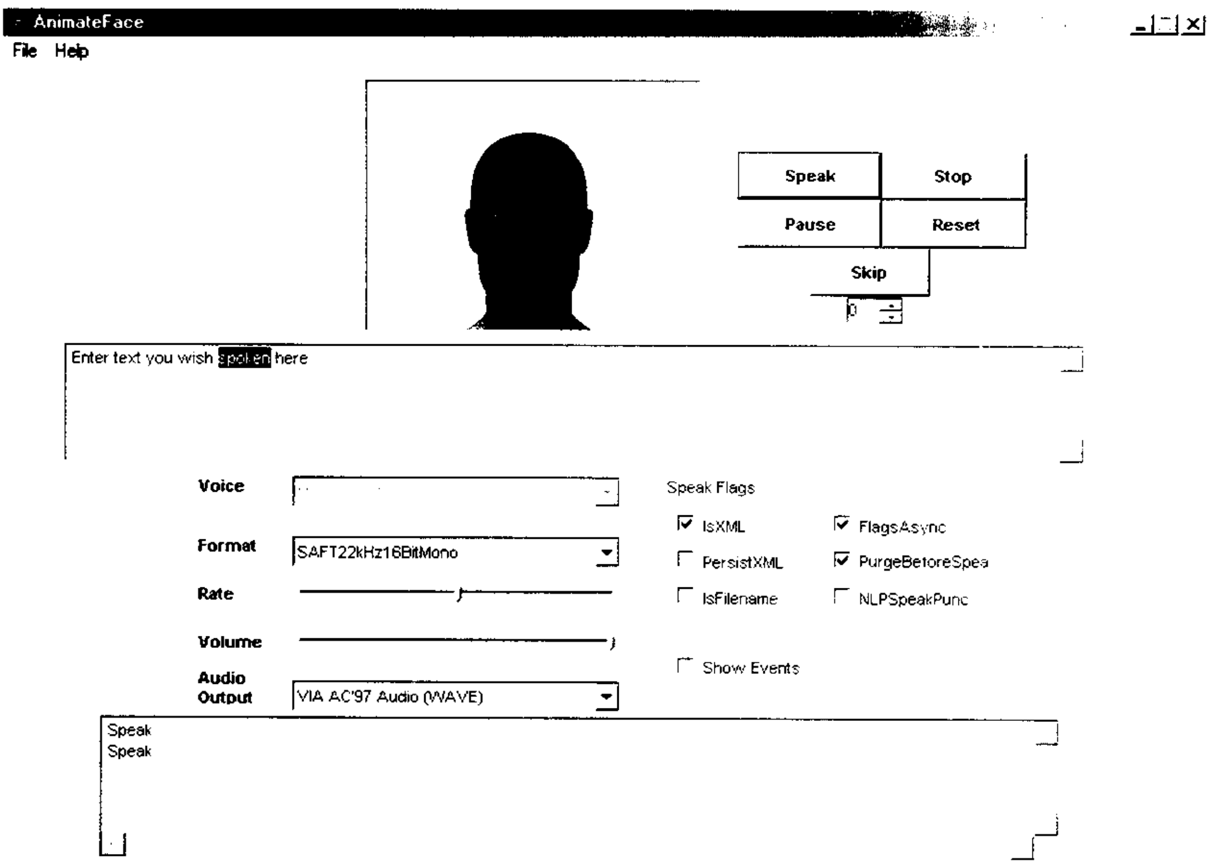
Volume

Audio Output: VIA AC'97 Audio (WAVE)

Speak Flags

- [✓] IsXML
- [✓] FlagsAsync
- [ ] PersistXML
- [✓] PurgeBeforeSpea
- [ ] IsFilename
- [ ] NLPSpeakPunc
- [ ] Show Events


AnimateFace
File Help
Speak
Stop
Pause
Reset
Skip
0
Enter text you wish to sp
Voic
Forn
Rate
Volu
Audi
Outp
Speak a Wave File
Look in: My Documents
History
Desktop
My Documents
My Computer
My Network P...
My eBooks
My Pictures
Visual Studio Projects
File name:
Files of type: Wave Files (*.wav)
Open as read-only
Open
Cancel



Enter text you wish spoken here

Voice

Format SAFT22kHz16BitMono

Rate

Volume

Audio Output VIA AC'97 Audio (WAVE)

Speak Flags

[✓] IsXML
[✓] FlagsAsync
[ ] PersistXML
[✓] PurgeBeforeSpea
[ ] IsFilename
[ ] NLPSpeakPunc
[ ] Show Events

```
Speak
Speak
```

REFERENCES

12. REFERENCES

- The Visual Basic .NET Programming Language, Microsoft .net Development series -Paul Vick.

- Programming Visual Basic .NET- Jesse Liberty, O'REILLY Publications.

- Introduction to Programming with Visual Basic .NET –Gary J. Bronson and David Rosenthal.

- The 3Ds MAX Quick Reference – Michele Bousquet.

- Scott A. King, Member, IEEE and Richard E. Parent, Member, IEEE Computer Society 'Creating Speech Synchronized Animation' – May-June 2005.

- Tsuhan Chen, 'Audio Visual Speech processing' – Lip Reading and Lip Synchronization.