An Object Oriented Articulatory Synthesizer for Windows

Luís Nuno Silva, António Teixeira, Francisco Vaz

Abstract - **The main motivation for this work was the production of speech synthesis with near natural quality using a modular application with a user-friendly interface. One of the most promising methods is the use of articulatory synthesis, which is a technique based on direct modeling of the human speech production process.**

A modular application will allow the integration of several models, improved and new, in a user-friendly interface, increasing the quality achieved. The main area of application of this work is the study and synthesis of European Portuguese.

Resumo **– A principal motivação para este trabalho foi a obtenção de síntese de voz de qualidade próxima do natural através de uma aplicação modular e de fácil utilização. Um dos caminhos mais promissores para atingir esse objectivo é através do uso de síntese articulatória, técnica que é baseada na modelação directa do processo de produção humano.**

A modularidade da aplicação, disponibilizando uma interface de fácil acesso e utilização, irá permitir a adaptação e criação de novos modelos de forma a aperfeiçoar a qualidade conseguida até aqui. O trabalho está vocacionado para o estudo e síntese do Português Europeu.

I. INTRODUCTION

Artificial speech has been mankind's dream for centuries. For a long time man has been trying to produce speech synthetically, but his search has not ceased yet and the results are far from perfect.

Scientific curiosity about speech and its aspects motivated an increase of research in this field. There is a need for artificial speech synthesis systems that can be used by human-machine interfaces. Although distinct, these two aspects are complementary and converge.

The availability of systems with speech interface will promote the social and cultural development of people with special needs, and will increase welfare of the others.

European Portuguese synthesis will preserve our cultural identity and will promote future research. Being nasality an area with many unanswered questions and being the Portuguese language cited usually in Phonetic literature [1] by its profusion of nasal phonemes we consider this area of great interest. It is also known that nasality of Portuguese nasal vowels differs from other languages, like the French, due to existence of nasality contours [2].

Motivated by the need to produce better synthetic speech for our language and human curiosity, an articulatory synthesizer is in development, since 1995, at our department/institute. As a result of previous work, the need for a Windows version, an adequate user interface, and better internal programming structure was noted. The use of Microsoft Windows will increase its diffusion and accessibility. The new version of the synthesizer, presented in this paper, is modular to facilitate the addition of new modules and functionalities, and improves of existing models.

Although developed in Microsoft Windows[1] its design allows it to migrate to other platforms. Classes were written using the standard C and C++ programming languages. The user interface is friendly and modular. For that we merged OpenGL[2] with the Windows environment. The purpose of the user-friendly interface in development covers the use on articulatory synthesis studies as well as in other scientific fields like Phonetics.

II. SYNTHESIS

There are several ways to synthesize speech. Among others we have Formant Synthesis, which models directly frequency response; and Concatenation Synthesis, that uses pre-recorded natural signal samples. Articulatory synthesis is based on the human speech production physiological model, being the most potentially satisfying method to produce high-quality synthetic speech.

A. Articulatory Synthesis

An articulatory synthesizer produces synthetic speech using physical, anatomic and physiological features modulating the human vocal tract. Features like larynx position, jaw opening, lips opening and protrusion, velum and tongue position, enable to create a close model and thus modulate the human vocal tract. The Articulatory Synthesizer can be split into the physiological anatomic model, and sound propagation model (acoustic model). The sound is produced when the acoustic model is excited. Depending on the excitation signal different sounds can be produced. Utterance depends not only on the anatomic and acoustic models but also on the excitation signal. Some features of this signal are inherent to the speaker and can vary in time.

Articulatory synthesis is explained in some detail in [3].

[1] Windows is a trade mark of Microsoft Corporation

[2] OpenGL is a registered mark of Silicon Graphics Inc

III. PROGRAMMING TECHNIQUES

To implement the synthesizer, object oriented programming was used. Several abstract classes, parameter transfer protocol rules, and data structures were designed. To separate synthesizer models from their controls and viewers, the Model-View-Controller (MVC) concept was adopted.

During development, some principles were adopted. For instance, each articulatory model must have its own controller, or set of controllers, and its viewers. It is important that the controller knows the potential of the model, i.e., how to deal with it. It is also important that the model can be drawn [4].

A. Base Classes

The main abstract classes were designated base classes. Using abstract classes helps modularization and runtime configuration of the synthesizer, making possible substitution of models. These classes define only the criteria and methods used. Derived classes specify the models making their implementations. These classes are responsible for the several kinds of models that can be implemented. After a careful analysis of the various sub-models of an articulatory synthesizer, three base classes for models were obtained: Anatomic Model, Source, and Acoustic Model. From application of the Model-View-Controller concept two other base classes were considered: Controller and Viewer.

In our implementation these base classes are virtual classes. Examples of base classes and implementations are presented on figure 1.

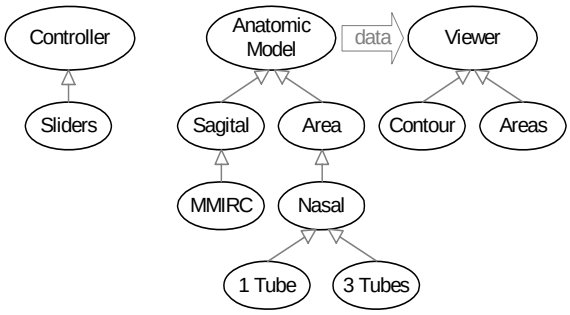

Figure 1 – Examples of base classes (Anatomic Model, Controller and Viewer) and derived classes.

Figure 2 shows how the main synthesis base classes work with each other. Acoustic Model gets area function information from the Anatomic Model, and a sample of the excitation from the Source Model to produce a speech sample. By continued repetition of this cycle, controlled by a Controller, a speech waveform is obtained.

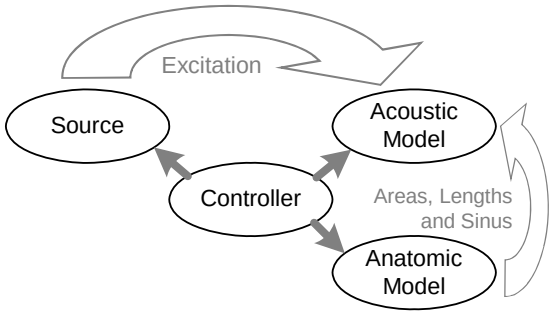

Figure 2 – Synthesis base classes

B. Controllers, Viewers and Models

As mentioned before, the Model-View-Controller concept was applied. Our main concern was to try to develop general controllers and viewers. In this way, when adding a new model the effort needed to also develop a controller and a viewer can be, in some cases, avoided.

This will raise the question: "*What is the use of general controllers and viewers?*". Sometimes it is not important the user interface of a model but the synthesis algorithm. Therefore there must be a way of using an anatomic model within any synthesis method. It is not simple to gather all models on a controller; thus, some controllers must be updated after a model addition. Usually general viewers show only the common features of all models. For example, all articulatory models produce areas and lengths, and these features can be presented in several ways.

There's still another question: "*How does general controllers work?*". General controllers don't know the model, they only know the synthesis process, but they can question the model about its needs. For example, each articulatory model has a different set of parameters, articulatory parameters, and each parameter has its own range. The model should have the ability of giving this information when the controller asks it.

B. 1. User Interface

Another important matter is the user interface. The user interface can be a specification of a controller class. This allows the existence of several kinds of interfaces, each one with different kinds of viewers. This special controller has the ability of control other controllers and viewers. A user action must be reflected on the application, i.e., the user is above any kind of control and it is him that controls the synthesizer.

C. Data Transfer Protocols

Some classes must be able to communicate. Data transfer rules are an important subject in this design, it is important to define rules that can be generalized.

On figure 3 examples of two parameter transfer rules are shown. First, it is shown how areas and lengths are defined and grouped for anatomic data transfer between models. The first position on the structure, n1, indicates the number of elemental tubes of the first section. Now, it is known that the next n1 elements have the values, ordered by tube, of some property of tubes belonging to that section. If the value stored on the n+1-th position (i.e., n2) is negative the structure stops there. If not there is another section with n2 tubes, and so on, until a negative number is found. This kind of structure allows an unlimited number of tubes and sections.

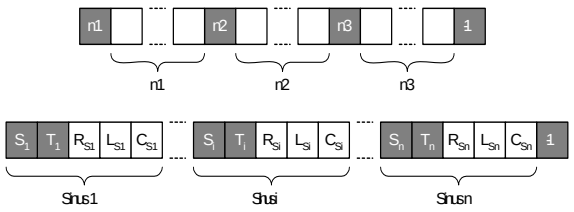

Figure 3 – Examples of parameter transfer rules. On top, how areas and lengths are defined and grouped for transfer of anatomic data between models; at the bottom, how information regarding sinus is transferred.

The second structure, used for transfer of information regarding sinus, has the same stopping rule, but in this case is known that each sinus is defined by five features: section and tube where it connects to, and RLC resonant circuit values.

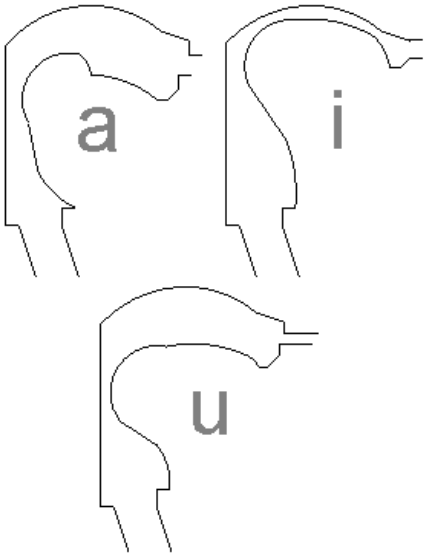

Figure 4 – Sagittal contour for 3 vowels represented using the MMIRC articulatory model.

IV. Models Already Implemented

In this section, derived classes representing the models implemented, re-implemented or adapted from the previous version of SAP (from the Portuguese *Sintetizador Articulatório para o Português*, in English *Articulatory Synthesizer for Portuguese)*, running in Linux, are presented.

B. Anatomic Model

We have implemented a sagittal two-dimensional articulatory model for the vocal tract and a comprehensive easily configurable nasal tract model (Figure 5).

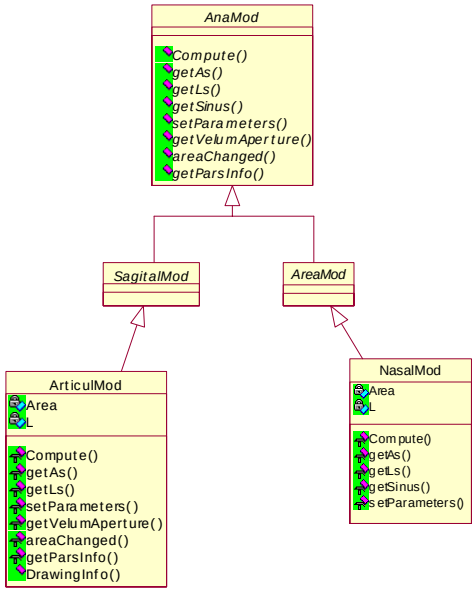

Figure 5 – Anatomic model class diagram. The ArticlMod class, implementing MMIRC sagittal vocal tract model, is a specialization of the SagitalMod general class. NasalMod class, developed for nasal tract modeling, derives from AreaMod generic class. At the root of this hierarchy is the virtual class AnaMod (Anatomic Model).

B.1. Vocal Tract Model

The anatomic model used assumes midsagittal plane symmetry. The output is an estimate of the vocal tract cross-sectional area [5], figure 4. Our model is an evolution [6, 7, 3] of the MMIRC (Mind Machine Interaction Research Center) model, which is a modified version of the Mermelstein model. It uses a non-regular grid to estimate section's areas and lengths. This process is described on figure 6.

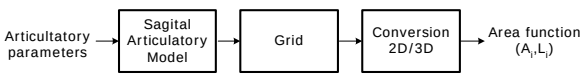

Figure 6 – Estimation of the cross-sectional area function from articulatory parameters.

The following class methods were implemented:

- setParameters – inputs of the model, in this case is a pointer to a structure with articulatory parameters;
- Compute – start the computation of the cross-sectional area function (figure 6);
- getAs and getLs – outputs of the model, pointers to structures with the areas and lengths of the split sections of the oral tract (figure 3);
- getVelumAperture – returns velum aperture (used when coupled with a nasal model);
- areaChanged – Boolean, indicate input changes;
- getParsInfo – returns its parameters information;
- DrawinfInfo – returns its own information in order to be drawn by a special viewer.

The last method (DrawingInfo) is not general, returns model dependent data. Only a special viewer class can use it properly.

This model doesn't implements sinus cavities so the implementation of getSinus method returns a NULL pointer.

B.2. Nasal Tract Model

The Nasal tract can be considered as a side branch of vocal tract. The velopharyngeal port controls the coupling between these two tracts [5].

The normal human noses have two nostrils, so in the model there should appear two channels to model the nose. If we consider that those channels are symmetrical (quasi-symmetrical profile) we can model it as single tract (1 tube). Otherwise modeling must be split in three: common tract, right and left nostril (3 tubes).

The implemented model, based on the proposal of Marilyn Chen (1997) [8], uses information from Dang and Honda's (1994) MRI measurements [9], assumes symmetry of nasal tract and includes Maxillary Sinus (figure 7).

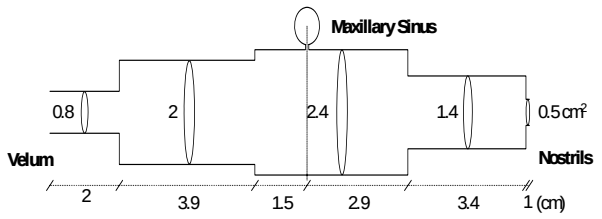

Figure 7 – Nasal model (Chen 1997) [8].

Due to its static nature, the configuration stays the same after definition. This is not exactly true because the coupling section depends on oral tract velum aperture, i.e., the first tube of the nasal tract should have the same size of the velum aperture. The coupling smoothness depends on the number of tubes used. Therefore the definition of the Nasal model is made on a file (see Annex A). Different nasal tract models can be easily defined by only changing the configuration file.

The following class methods were implemented:

- setParameters – inputs of the model, is a pointer to a structure with the velum aperture of the oral model, previously computed;
- Compute – computes the area of the non-fixed tubes, coupling between Nasal and Oral tract;
- getSinus – output of the model, pointer to a structure with sinuses values (figure 3).

C. Acoustic Model

The acoustic model is responsible for speech wave generation. The output of main synthesis base classes is a sound wave.

The impulse response is given by the Inverse Fourier Transform (IFFT) of the acoustic transfer function of a given vocal tract configuration. We used the fast implementation of the FFT developed by M. Frigo and S. Johnson (1999) [11]. The convolution of the impulse response with the glottal excitation signal will produce the sound. A frequency domain analysis and time domain synthesis method – usually designated as the hybrid method [10] – is used.

Before any kind of computation, the acoustic model must know the configuration of anatomic models (vocal and nasal). Pointers to oral and nasal tract anatomic models must be passed to objects of this class, to make possible retrieving areas, lengths, and sinus data.

In a general oral and nasal tract configuration, speech radiation can occur at several points, like lips and nostrils. Making the model more flexible, facilities for selecting which of this radiation points are taken in account in the synthesis process were implemented. As an example, speech radiated only at the nostrils can be obtained by appropriate choice of configuration parameters.

All these features can be set and accessed using seven methods only:

- setParameters – allow radiant sections contribution for synthesis definition;
- setArticMod – Anatomic model pointer, allow Anatomic model's areas, lengths and sinus access;
- setNasalMod – same as previous, regarding the Nasal model;
- getTrackLoad – returns vocal tract load, needed by the exciter;
- FirstConfig – reset the model and compute the first static configuration (compute the impulse response only);
- NextConfig – model reconfiguration, reset the model and computes a new impulse response;
- NextSample – sound sample production by the convolution of the impulse response with samples of the glottal excitation signal.

C.1. Impulse Response Computation

The frequency response calculation is implemented by a protected method, Synthesize, of the acoustic model class.

This method is complex; the first time it is used it will build the tract model structure that will only be changed when there are any changes on the anatomic models.

To build the tract model structure, i.e., the tubes structure, it is necessary to know the number of sections and tubes of each model. This information can be obtained from the transfer data rules used by anatomic models. In subsequent stages the model must check the tube structure, changing it if needed, and compute again the transmission matrix ABCD [3] of each tube. This task is performed by the acoustic model class's method `Tract`.

Tract method defines both vocal and nasal tract data structure. Information obtained from the anatomic models is stored in a structure (figure 8) capable of dealing with multiple and variable number of sections and radiating points. It is similar to both kinds of tract: vocal and nasal.

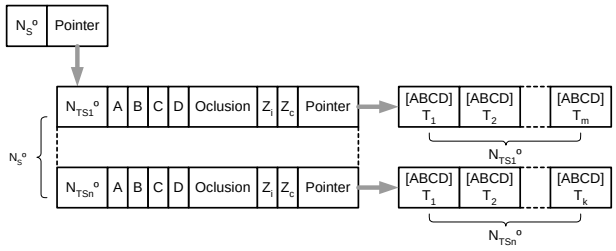

Figure 8 – Acoustic Model Data Structure.

The structure header is a `TractSize` variable that contains the tract's number of sections and a pointer to a `SectionDef` array. Each element of this array is a structure that contains the number of tubes of that section, the transmission matrix ABCD, section occlusion flag, input and load impedance and a pointer to a `CTube` array.

C.1. Tube model class (CTube)

CTube class creates and manages elemental tube model. From tube's area and length it is possible to compute its ABCD matrix. If the area is too small the occlusion flag is activated. Each tube can be associated with a sinus; its impedance will reflect this association. Each tube has a load, which can be another tube or the radiant impedance. The ABCD matrix and impedance are a function of frequency; so all class returned values are frequency dependent.

CTube's methods are:

- `set` – input parameters, can be tube's area and length or a coupled sinus parameter;
- `get` – returns the computed ABCD matrix;
- `compute` – computes ABCD matrix based on input parameters;
- `Zrad` – computes and retrieves radiant impedance, supposing that is a radiant tube;
- `Zoc` – computes and retrieves load impedance, supposing it is an occlusion contiguous tube;
- `hasSinus` – Boolean, verifies the existence of coupled sinus;
- `Zsinus` – computes and retrieves load impedance of the coupled sinus.

Methods like get, compute, Zrad, Zoc, and Zsinus return computation results for a single frequency, referred as an input parameter.

D. Source

To obtain glottal excitation, $u_g(t)$, it is necessary to model several subsystems involved: lungs, subglottal cavities (under the vocal chords), the glottis and supraglottal tract (cavities above vocal chords). The scheme on figure 9 represents all these subsystems.

The role of the lungs is the production a quasi-constant pressure source, represented on the model, figure 10, by a pulmonary pressure source p_p in series with the resistance R_p.

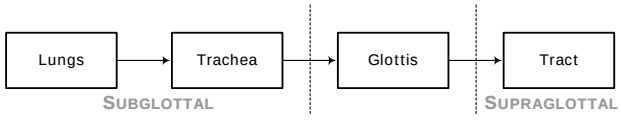

Figure 9 – Subsystems involved on glottal excitation calculation.

To represent the subglottal region, trachea included, we use the approach of Ananthapadmanabha and Fant (1982), i.e., we use three RLC resonant circuits [12].

Prado's approach, using a direct parameterization of the two mass model for glottal areas, was the choice to model vocal chords [13].

Systems above glottis, the tract, can be modeled by an input impedance $z_e(t)$ – or the pressure $p_{supra}(t)$ obtained from $z_e(t)$ and glottal flow convolution – or cascading RLC circuits. The input impedance, obtained from Acoustic model, allowing a better modeling of frequency depending losses, was chosen. More details about this glottal source model can be found in [3] and [7].

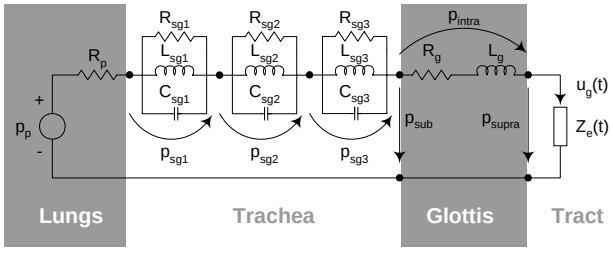

Figure 10 – Electrical analogue for glottal excitation calculation. Adapted from [3] and [7].

Class Exec01, implementing the described glottal source, is a specialization of the Exciter base class (figure 11). It can be configured as interactive or non-interactive. Non-interactive considers a null vocal tract load.

This source model is controlled by two kinds of parameters: time variant and time invariant. The first ones are used to control speech utterance and naturalness (see

Annex B). Time invariant (see Annex C) are set using a configuration file (see Annex D).

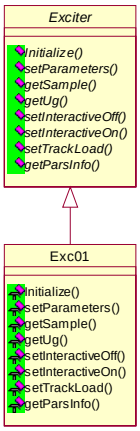

Figure 11 – Exciter model class diagram. Methods of base abstract class and methods implemented in the derived class Exc01 are shown.

Class methods implemented are:

- Initialize – set and reset model, time invariant parameters;
- setParameters – inputs of the model, is a pointer to a structure with the time invariant parameters and samples in a period;
- setTrackLoad – set vocal tract load into the model;
- setInteractiveOn and setInteractiveOff – enables or disables the interaction with vocal tract;
- getSample – returns glottal flow derivate of a given time instant;
- getUg – returns glottal flow sample;
- getParsInfo – return its parameters information.

V. CONTROLLERS

Controller base class allows the existence of several controls to perform the same or different tasks. The user can choose the best controller that suites the application. The user interface can also be considered as a controller, or the main controller.

As an example, an articulatory model controller class diagram can be seen on figure 12. The controller knows the model (pointed by m_pModel) and can control a set of viewers (m_pViewer).

On the base class we have the abstract methods:

- Initialize – set the sample rate and the number of samples used in signal processing and mathematical operations;
- Compute – starts the process, i.e., starts the control procedure;
- setParamsMod – Parameter model pointer, allow a coherent parameter transfer;
- setArticMod – Anatomic model pointer, allow the use/control of the Anatomic model;
- setNasalMod – same as previous, regarding the Nasal model;
- setExcMod – Exciter model pointer, allow the use/control of the Exciter model;
- setAcusMod – same as previous, regarding the Acoustic model;
- setSound – Sound model pointer, allow the use/control of the Sound model;
- setViewer – same as previous, regarding the Viewer model;
- addViewer and delViewer – allows the addition an removal of extra viewers;
- setControl – allows the existence of an alternative control model.

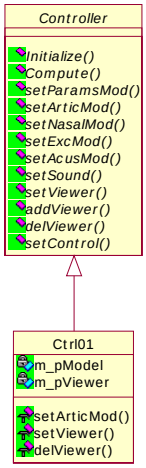

Figure 12 – Controller model class diagram.

VI. TIME VARYING PARAMETERS MANAGEMENT

In general, parameters for source and anatomic models vary over time. In many situations each parameter varies independently of the others.

To support and manage time varying parameters and make possible independent control of each parameter, a special class, named Parameters, was created. This class reads and stores parameters data from and to files; inserts, adds and removes Targets[3] (value of a parameter at a specific time); makes data interpolation and automatically configures itself depending on the models used.

In its implementation, this class, to make possible obtaining a parameter value at a specific time using parameter name directly, uses a map (class from the Standard Library [14]) object, for all parameters, indexed on parameter name.

On class constructer there is a reference of the anatomic and source models. It is then asked to those models for the name, file storage and range (maximum and minimum) of the parameters they use. Anatomic and source models have a method to return that information, getParsInfo. This way, parameters will automatically be configured according to the model used. When needed there is a

[3] A class with this name was developed to deal with time-value pairs.

method, getInfo, that returns a list with all the information regarding the parameters used.

Storage is not made on an individual file, but in several located on a folder. All operations are folder based; the model defines the name of the files. The save and load methods are used to store and get parameters data from a folder, they both have the folder name as argument.

Addition of targets is made by the setValue method, which has the name of the parameter and its value on a given time instant as argument. Removal of targets can be made in two ways: value removal (eraseValue) or removal of all defined targets (eraseAll). First one needs parameter name and the time instant as input, the other only needs the parameter name.

To obtain a value for any time instant linear interpolation is used. getValues method returns a value for a given time instant.

VII. VIEWERS

Several viewers were implemented, specially for anatomic models. OpenGL was particularly useful in viewer implementation.

The two-dimensional sagittal representation of the MMIRC anatomic model, an example of a model dependent viewer, can be seen at the left top corner of figure 16, in the next section.

An example of a general viewer implemented was the area function viewer for anatomic models. This viewer, showing cross-sectional area as function of position, can be used with any anatomic model capable of producing the so-called area function as output, which is usually the case.

VIII. APPLICATION

To use the developed models, an application capable of synthesize speech segments from parameter sequences was created. These sequences can be defined, changed (edited) or used to produce synthetic speech. The synthesis status is presented step by step on a graphical interface and, finally the resulting sound is played.

A. Modeling

More classes were needed to set up the application. The synthesizer models were already developed, so we need to define some control and viewer classes. To control two classes were implemented: MainControl class, strongly connected to the user interface; and CtrlPitchSync, that controls the synthesis process.

Also two kinds of user interface were implemented. One is a viewer that shows only synthesis information; the other, using dialog boxes, is an interface with the controllers.

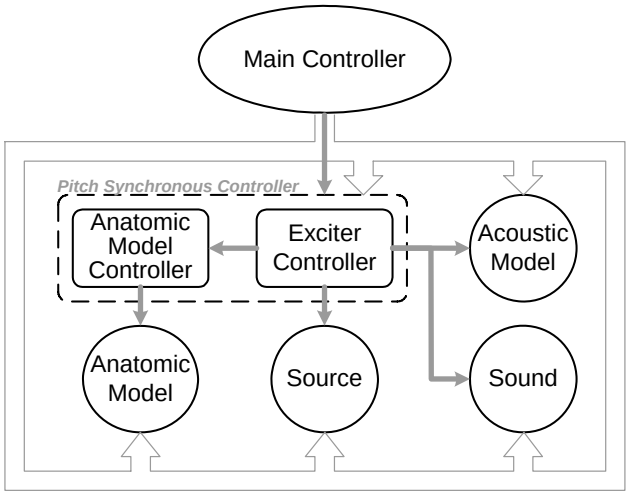

Figure 13 – Application model, showing Main Controller, Pitch Synchronous Controller and main objects responsible by modeling anatomic configuration (Anatomic Model), sound generation (Source), sound propagation and radiation (Acoustic Model), and sound handling (Sound). Main Controller creates all the other objects, action represented by unfilled arrows. The Anatomic Model Controller part of the Pitch Synchronous Controller controls directly the Anatomic Model, Exciter Controller controls Source, Acoustic Model, Sound and also the Anatomic Model Controller.

A.1. Main Controller

When the application starts up it creates a controller, the main controller. It has the ability of creating all the synthesizer structure, i.e., it creates all the objects within the synthesis process or that will be used by the synthesis process. Only it has the power of creating and destroying objects; however, other classes can ask to do it.

After being created objects need to be initialized before being used. This initialization shouldn't be shared with alternative controllers (synthesis process controllers). The initialization is sometimes made using files that can be changed through the user interface, however there is always a default initialization.

Figure 13 shows the application synthesis process classes, with the exception of Parameters class. The filled arrows represent the control flow; the others represent the creation and initialization process.

After creating and initializing everything needed, the main controller can ask the alternative controller to start the synthesis process.

A.2. Alternative Controller – Pitch Synchronous

Pitch Synchronous is an alternative controller that will control the synthesis process. Synthesis process can be either static (time invariant system parameters) or dynamic. The Pitch Synchronous is a dynamic synthesis process, where there are no changes on the anatomic model during a period of glottal excitation signal, T_0.

This period is in general non-regular, depending on the time variation of the fundamental frequency (F_0) and Jitter:

$$F_0 = F_0\left(1 + 2 \cdot U(-0.5,0.5) \cdot \frac{\text{Jitter}}{100}\right) = T_0^{-1}$$

where U(-0.5, 0.5) returns a uniform random number on the range [-0.5, 0.5). The resulting pitch period is rounded to the nearest sample rate multiple, in order to simplify the operations and maintain the synchrony. The number of samples in a period is given by the ratio between the sample and fundamental frequencies.

Then, for each sample, the glottal excitation value is computed that will operate in the acoustic model to produce a sound sample (figure 14) and this sample will be sent to the Sound model buffer.

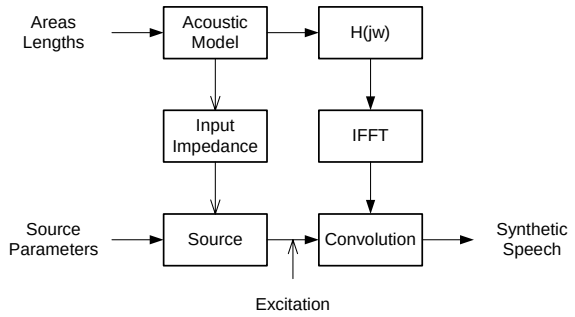

Figure 14 – Information flow for each excitation period. Area function (Areas and Lengths) from the Anatomic Models is used by the Acoustic Model to obtain H(jw), at several frequencies, needed to impulse response calculation, by application of an IFFT; also based in the area function Acoustic Model calculates tract Input Impedance used by Source Model to calculate a sample of the glottal excitation wave; finally speech is obtained by convolution. In Pitch Synchronous mode, source parameters are considered fixed during a pitch period, and H(jw) is calculated only at the beginning of a period.

Initially synthesis time is zero and it is increased by the consecutive excitation periods. In each period it is asked to Parameter model to evaluate the parameters for that synthesis time. All values will be re-computed based on the new parameters values [15], repeating the process until a predefined – user configurable – simulation time is reached.

A.3. Sequences Definition

Before starting the synthesis process, the user must define a sequence. This definition can be completely new or use stored values (load) (figure 15).

After sequence definition, there is a tree control that shows all the parameters that will be used. That tree control has two main branches, the edit and views. While we can edit only one parameter, it is possible to view (colored graphical view) five parameter sequences simultaneously.

In addition to the graphical representation, the parameter under edition is also shown on a table. If the values of the table are selected by clicking the mouse it is possible to update or remove them. By default the edit area adds values to the original set, however, if there is a previous value for the same time instant it will ask the user to confirm the update.

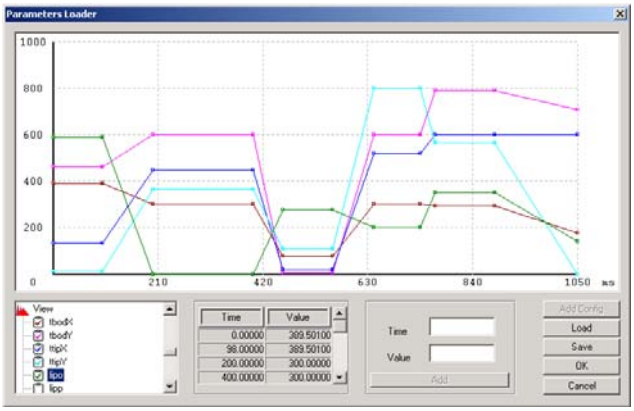

Figure 15 – Sequence Editor. Main area show, in different colors, variation over time of selected articulators; left bottom zone can be used by the user to configure what parameters are displayed and editable; bottom center zone helps in target edition and removal; right bottom zone contains buttons to load and save parameter sequences.

On future versions the button Add Config will allow sequence concatenation/merging, making possible the addition of stored sequences to the current editor sequence.

The save button will save precious time to the user on the next time he wants to use the same sequence.

B. User interface

The aim of the user interface is to describe all synthesis process and show some of the signals. Beyond the relevance of these signals for research, the signals can be very helpful for teaching.

Simultaneously representation of articulatory and acoustic information can be very useful for research.

In articulatory and acoustic phonetics teaching it would be very interesting for students to see the articulators effects on the areas, excitation and spectrogram. On speech therapy area function information and the spectrogram can be a precious help to those that cannot listen.

The interface has two parts (figure 16).

Articulators can be represented in a form of a sagittal contour of the vocal tract, the most common way of speech articulators representation. Using this representation, the time evolution of vocal tract can be presented as successive shapes in different time instants. Due to the synthesis process slowness, only the last four representations of the sagittal contour are presented. A time colormap is presented below the contour, with white representing the last contour.

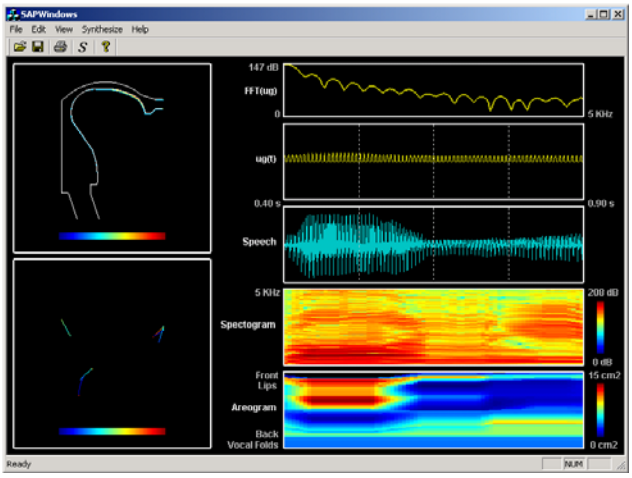

Figure 16 – Visualization of the synthesis process in the developed articulatory synthesizer, SAPWindows.

Below sagittal contour representation we can find the time space evolution of three articulators: lips, velum and hyoid. This representation also obeys to a colormap, presented below. Each color represents a time slot, in the end it will jump to the first color of the colormap. With this representation it is possible to track these articulators evolution.

On the second part representations of speech and glottal excitation (flow through vocal chords) signals can be found. First represents the Fourier Transform of the excitation signal during one period. Second represents time evolution of the source signal, ug(t). Third one represents the speech signal.

An important representation of a signal is its power spectrum through time. It is a representation of the frequencies energy, represented by a colormap, along time and is usually known as spectrogram.

Having the cross sectional areas of all synthesis process it is possible to have its variation over time using a representation similar to the spectrogram. Because this kind of display visualizes vocal tract cross sectional areas is called an areogram, by analogy with the spectrogram where the spectra are displayed as a function of time [16].

IX. Conclusions

The main result of this work is a modular articulatory synthesizer architecture, using oriented object, a new approach on the field. This will make new model addition and handling easier, enabling better quality to future synthesizers. The effort of European Portuguese synthesis is a credit for our social-cultural development.

Although no user tests were done this application was shown to potential users, in linguistics, and they think that, even in its present incomplete state, it is useful in phonetics teaching. It also can be very useful to teach Portuguese as a foreign language.

This is an important step, but the development of this synthesizer is an unfinished task and more work should be done to create new models and improve the already implemented.

Acknowledgments

This work was supported, in part, by project PRAXIS/P/PLP/11222/1998, *Síntese Articulatória do Português*. Authors thank: Beatriz Sousa Santos for her collaboration in the visualization part of the articulatory synthesizer GUI; developers of the FFTW for making this library available; MMIRC, University of Florida, for all the information needed for models' implementation; Lurdes Moutinho from Languages and Cultures Department, University of Aveiro, her remarks regarding SAPWindows applicability in Phonetics teaching; António Branco his contribution to articulatory model implementation and first steps toward a modular object-oriented implementation; the several Windows programmers that have contributed to solve the many problems encountered.

Annexes

A. Nasal model definition file

```
% These lines are used to comment this file
% Each line should have up to 198 characters
%
% #1 #2 #3 #4, where:
% #1, is the number of tubes on the common ↵
section of Nasal Tract;
% #2, is the number of tubes on the right ↵
section of Nasal Tract;
% #3, is the number of tubes on the left ↵
section of Nasal Tract;
% #4, is the number of sinus
6 0 0 1
% Number of the first fixed tube on the ↵
common section of the Nasal Tract
2
% Data:
% Ls As
2.0 0.8
3.9 2.0
1.5 2.4
-1 1.1 3.42e-3 29.7e-6    % (-1) sinus ↵
definition. After (-1) we have Rs, Ls and ↵
Cs values
2.9 2.4
3.4 1.4
1.0 0.5
%% (%%) End of data. All information ↵
written after this sign will be ignored.
```

B. Time dependent excitation model parameters

Parameter	Description	Typical value	Unit	*setParameters*
p_p	Pulmonary pressure	10000	*dyne/cm²*	lungs
F_0	Fundamental frequency (Pitch)	100-200	*Hz*	F0
OQ	Open Quotient	60	*% of T_0*	open
SQ	Speed Quotient	2		speed
A_{g0}	Minimum glottal area value	0	*cm²*	ag0
A_{gmax}	Maximum glottal area value	0.3	*cm²*	agmax
$area_1$	Glottal section area value	-	*cm²*	area
A_2-A_1		0.03	*cm*	slope
Jitter	F_0 random variation	2	*%*	jitter
Shimmer	A_{gmax} random variation	5	*%*	shimmer
Asp	Aspiration			asp

C. Time independent excitation model parameters

Parameter	Description	Typical value	Unit
R_p	Pulmonary resistance	8	*Ω cgs*
Φ	A_{g1} and A_{g2} phase difference	45	*Degree*
d_1	m_1 thickness	0.25	*cm*
d_2	m_2 thickness	0.05	*cm*
l_g	Glottis length	1.4	*cm*
k_1		1.37	
k_2		0.3	
ρ	Air density	1.14×10^{-3}	*g/cm³*
μ	Air viscosity	1.86×10^{-4}	*dyne·s/cm²*

D. Source model configuration file

```
# Definition of Interactive Source
#  (c) Antonio Teixeira, 1998
# dg [Koizumi oct 1987 ,pg 1185] in cm
0.3
# d1
0.25
# d2
0.05
# lg [Koizumi oct 1987 ,pg 1185]
1.4
# miu dyn.s/cm2
0.000186
# rho g/cm3
0.00114
# k1 e k2
1.37
0.3
# Lungs Resistance (cgs Ohm)
8
# subglottal(from Ishizaka et al, JASA, 1976)
# Freq (Hz), Bw (Hz), Indutance (mH)
615.0  246.0 3.8
1355.0 155.0 0.72
2110.0 140.0 0.27
# phase diference between Ag1 and Ag2
45
# Two mass parametrical model [Prado 91]
# 1= Titze area model
# 2= Two mass parametrical model [Prado 91]
2
# use Lglotal in calculations
1
# don't use Derivative of Lglotal
0
# don't use Viscosity of Glottis
1
```

REFERENCES

[1] J. Laver, "*Principles of Phonetics*", Cambridge University Press, 1994

[2] R. L. Trigo, "The inherent structure of nasal segments", In M. K. Huffman and R. A. Krakow, Eds, "*Nasals, Nasalization, and the Velum*", pp. 369-400, Academic Press Inc, 1993

[3] A. Teixeira, L. N. Silva and F. Vaz, *Síntese Articulatória: Uma Introdução,* Revista do Departamento de Electrónica e Telecomunicações, (neste número), 2002 .

[4] Dave Collins, *"Designing Object-Oriented User Interfaces"*, Benjamin/Cummings Publishing Company, Inc, 1995

[5] Donald G. Childers, *"Speech Processing and Synthesis Toolboxes"*, John Wiley & Sons, 2000

[6] A. Branco, *"Representação Visual do Modelo Articulatório para o estudo da Produção da Fala"*, Tese de Mestrado, Universidade de Aveiro, 1997

[7] António Teixeira, *"Síntese Articulatória das Vogais Nasais do Português Europeu"*, Tese de Doutoramento, Universidade de Aveiro, 2000.

[8] Marylin Y. Chen, *Acoustic correlates of english and french nasalized vowels*, Journal of the Acoustical Society of America, JASA, vol. 102(4), pp 2360-2370, 1997

[9] J. Dang and K. Honda, *MRI measurements and acoustic of the nasal and paranasal cavities*, Journal of the Acoustical Society of America, JASA, vol. 94(3 Pt.2), pp 1765. September 1994.

[10] M. M. Sondhi and J. Schroeter, A Hybrid Time-Frequency Domain Articulatory Speech Synthesizer, IEEE Trans. Acoustics, Speech and Signal Processing. ASSP, vol. 35(7), pp 955-967, July 1987.

[11] Matteo Frigo e Steven G. Johnson, "*FFTW User's Manual*", Massachussetts Institute of Technology, 1999.
URL: http://www.fftw.org/ (March 2001)

[12] T. V. Ananthapadmanabha and G. Fant, *Calculation of true glottal flow and its components*, Speech Communication, vol. 1, pp 167-187, 1982

[13] Pedro P. L. Prado, *"A Target-Based Articulatory Synthesizer"*, PhD Thesis, University of Florida, 1991

[14] J. Weidl, *The Standard template Library Tutorial*, Information Systems Institute, Distributed Systems Department, Technical university Vienna, 1996.

[15] Luís Nuno Silva, *"Desenvolvimento de um Sintetizador Articulatório"*, Dissertação de Mestrado, Universidade de Aveiro, 2001

[16] Sorin Dusan, *"Statistical Estimation of Articulatory Trajectories from the Speech Signal Using Dynamical and Phonological Constraints"*, PhD Thesis, Dept. of Electrical and Computer Engineering, University of Waterloo, Waterloo, Canada, 2000
URL: http://crg7.uwaterloo.ca/~sdusan/publications.html (May 2000)