A Component-Based Framework for the Development of Virtual Musical Instruments Based on Physical Modeling

Panagiotis Tzevelekos, Thanassis Perperis, Varvara Kyritsi and Georgios Kouroupetroglou
National Kapodistrian University of Athens, Department of Informatics and Telecommunications, Athens, Greece,
{taktzev, a.perperis, koupe}@di.uoa.gr

Abstract* — *We propose a framework for the design and development of component-based woodwind virtual instruments. Each functional part of the instrument is represented with an independent component, and can be created with different approaches, by unfamiliar constructors. Using the aforementioned framework, Virtual Zournas is implemented. The user can experiment with the instrument, changing its physical properties. Instrument control is performed via MIDI files or external MIDI devices.

I. Introduction

Music performers are always in search for greater possibilities in terms of artistic creation and expression. Such needs led to the development of Virtual Musical Instruments (VMIs). Even though they were initially designed for live performances, VMIs can find use in a plethora of applications, spreading from educational programs to virtual museums [1], [2], [3].

Until today, there have been few strictly predefined and autonomous frameworks for developing VMIs [4], [5]. Researchers study VMIs from different points of view, focusing either on the sound synthesis procedure [6], [7], or on innovative means of gestural control and interaction interfaces [8], [9], [10]. Educational and "museum" applications center mainly on the photorealistic 3-dimensional visual representation of the instrument [2], [11]. Moreover, such systems do not permit the use of different methodologies for sound synthesis and visualization, and they don't predict for a life cycle with independent designers and users, able to alter an existing VMI or build one from scratch.

In this paper, we present a framework for the design and implementation of VMIs based on software components, which allows the collaboration among different developers with different approaches. We begin by analyzing sound synthesis by physical modeling and some reasons why we find it to be the most appropriate technique to use with VMIs. After showing how every musical instrument is composed of different functional parts (i.e. excitation, oscillating body, resonance body), which play their own roles in the sound production procedure, we proceed with the description of the framework. Each functional part of the instrument is represented with an independent component. Such components are linked together in order for the complete VMI to be constructed. Based on the proposed framework, Virtual Zournas is implemented. The appropriate components and a full use-case scenario are presented.

II. Physical Modeling

Physical models are based on mathematical models that can describe the physical acoustics of a real-world instrument. By describing its acoustical behavior with equations, we understand it better and we can simulate it better. As C. Roads states [12]: "*a physical model embodies the Newtonian ideal of a precise mathematical model of a complicated mechano-acoustical process*". That is, if it is likely to collect all the equations corresponding to sound generation and propagation and render them by computing means, then the sound output would have great resemblance with the one of the real instrument.

It should be made clear however, that the main purpose of building physical models of musical instruments is not to simply replicate them. VMIs based on physical modeling provide the user with the ability to control them in a straightforward manner during the sound generation process, as well as during a live performance.

Physical models are often described as musical "reality generators", since they can develop forms that have nothing to do with reality. Thus, the real world can serve merely to inspire the creation of surreal sounds and instruments [12], [13].

The sound production process is quite similar in all acoustic instruments. An *excitation* sets an instrument part, the *oscillating body*, in periodic motion. The oscillation sets in motion other parts of the instrument, mostly described by the term "*resonance body*". Finally, the secondary oscillations produce acoustic waves that travel through air and reach our ears.

While analyzing the acoustic behavior of a musical instrument, attention should be given to all of its parts, even to those whose role in sound production is thought to be less profound or evident. Slides and valves in brass instruments, keys and tone-holes in woodwinds, the bridge in strings, are such examples.

It is hence obvious, that a musical instrument can be considered to have different functional parts, each one having a different task in the sound production process. In this way, in order to design a physical model, it is a common procedure to focus on each of these parts separately. One must derive the equations that describe the acoustics of each part, build individual models for them and finally connect them to form the complete instrument.

It should also be possible to substitute the model of a part without having to change the model of the whole instrument, in the same way that a clarinet player can substitute the embouchure of his clarinet with having to buy a new clarinet. Such options are valid from the manufacturer's point of view as well. A guitar maker could add an additional string, without putting up the entire guitar from the beginning.

In Fig. 1, examples of instrument parts and their acoustic behavior are given. By combining parts from each class, a well-known or even an original musical instrument occurs. Originality refers not only to the form of the instrument, as in the case of the saxophone introduction, but also to new ways of performing classical instruments. Performance is considered to be a part of the instrument itself, since it is strongly related with the excitation mechanism and the use of other parts. In fact, we should consider an instrument functional part as an acoustic procedure, not just as a material piece of the instrument.

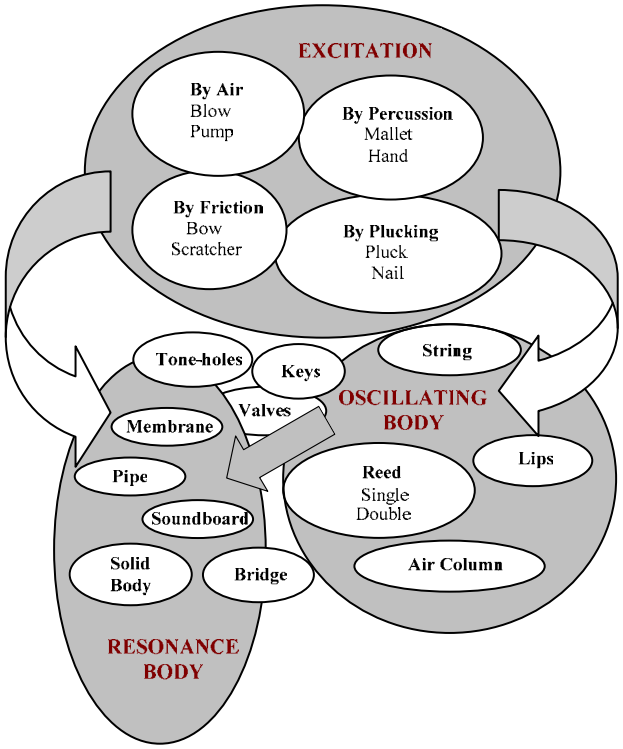

Fig. 1. Functional parts of musical instruments

III. THE KTISIVIOS FRAMEWORK

A. *VMIs*

We define Virtual Musical Instruments (VMIs) as software applications that provide users with the means necessary to control, fine-tune, experiment and perform music with an as intuitive and natural way as possible. The basic parts of a VMI are a sound synthesis engine or synthesizer and a graphical user interface or a hardware control device.

The music software industry has responded to user needs by providing a number of computer applications that can be used to develop, amongst other, VMI applications. The most popular of these applications use the famous visual patching paradigm to allow users to create synthesizers [14], [15], [5]. Creating a musical instrument in such environments involves fitting patches together in a graphical manner, mimicking hardware assemblage. Another popular approach involves using high-level musical languages that allow sound synthesis and control through programming [16]. Both of these approaches have proven very successful by assigning development responsibilities on the users.

There are however groups of users that are outside the scope of such applications; users that lack or should not have to possess expertise with programming, knowledge of sound physics or familiarity with digital signal processing; users that are profoundly involved in the music domain and should not be directly involved in the development process. We classify musical instruments craftsmen, performers, educators, musicologists and music composers as example groups of genuine "music" users.

These groups of users have to use a software music environment whenever they want to operate a VMI, without having direct access to the VMI as an independent application. Loading scripts, projects or configuration files, and in general, interacting with a VMI inside a software tool or environment forces the user to gain familiarity with the environment.

This demand for familiarity and, ultimately, usage competence, is often troublesome and acts as an obstacle and a deficiency factor for using a VMI in a natural and intuitive way. Even more, frustrating situations occur, provided that a user can handle a VMI as a developer, giving her full rights to manipulate source code and design of the VMI and the ability to potentially "break" it. The non-developer, "music" user should be protected from such situations and interface with the VMI directly as a rigid, compact entity.

As far as software development in the music software industry is concerned, the dominant issues of reusability, maintenance and ultimately cost-effectiveness, are present. Component based development [17], as an approach that successfully deals with such issues, is partially used in the music software domain. There are some add-on modular approaches, but still, these "expansion" modules are far from autonomous and depend heavily on being hosted by some application. It is definitely not possible to collect, compose and finally produce a complete application using such modules. Thus, the overbearing pattern of proprietary monolithic applications still applies in the music software domain, leading to an increased cost of products, as a result of poor reusability, backbreaking maintenance and evolution of software.

The choice of physical modeling as a sound synthesis technique is imperative, once someone decides to provide highly efficient, realistic sound synthesis. Although some software applications endorse physical modeling as a sound synthesis technique, none is devoted to it.

We decide to take a different approach that will provide a clear distinction between users and developers. We present KTISIVIOS, a component-based framework that can be used for developing custom-made VMIs based on

physical modeling. We propose a development life cycle for KTISIVIOS that handles the distinction between users and developers in an effective, concise way. We move physical modeling a step beyond, by decomposing the holistic model of a single instrument into small, independent models of the parts of the instrument that interconnect and are replaceable, just as in the case of a real world musical instrument.

B. Framework Description

In this section we present KTISIVIOS, a component based framework for the design and development of VMIs. KTISIVIOS provides guidelines and tools for fast and costless implementation of flexible and optimized VMIs. Digital sound synthesis using physical modeling and musical instrument parts representation through software components constitute the main philosophy behind the framework. KTISIVIOS has an intrinsic *object oriented* philosophy and is implemented in C# using the Microsoft .Net framework.

KTISIVIOS framework is developed according to specific requirements. The most important are: to be open regarding the number, type and control means of implemented VMIs, to allow the end-user to modify in real time the values of physical properties of the instrument represented by the VMI (e.g. shape and dimensions), with corresponding sound result, and to give the application the prospect to run distributed, that is to run on more than one computers, which communicate through network.

We find KTISIVIOS to be a useful tool for a large user group, from common users who desire to exploit the musical potentials of computers to expert users with specific needs. Such expert users are researchers attempting to reconstruct antique or traditional musical instruments, contemporary performers, composers trying to produce original timbres, craftsmen and musical instrument manufacturers willing to hear the sound that a musical instrument produces before building it and students using it as a music education tool.

The main feature of our framework is the modular design through component based architecture. Following the standardizations and specifications of KTISIVIOS, developers implement software components that simulate the acoustic behavior of independent musical instrument parts. It is also possible for a component to apply common liturgical functions for the VMI itself, such as load, save, help and exit functions.

KTISIVIOS implements interfaces for each component's structural parts. In object oriented programming, an *Interface* is a *reference* type and it contains only *abstract* members. Interface's members can be *Events*, *Methods*, *Properties* etc. But the interface contains only declarations for its members. Any implementation must be placed in the class that realizes them. Thus, component development becomes easier and more specific. The overall structure of every component is strictly specified and the developer (see III.*C*) must only inherit his classes from these interfaces and realize the abstract members according to his requirements.

KTISIVIOS states that every VMI component is structured from the following parts:

- Graphical interface
- Controller
- Physical model.

In Fig. 2 we present the arrangement of these parts in a VMI component.

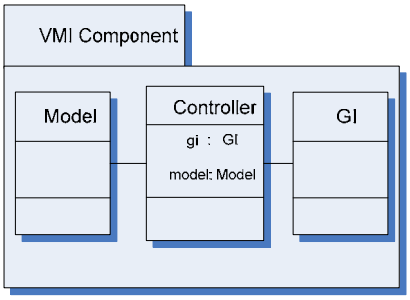

Fig. 2. VMI Component and the arrangement of the internal parts

Physical Model (Model) implements the real time algorithms that solve the mathematical equations, which describe the acoustical behavior of the corresponding part of the instrument. This element of the component is responsible for the digital sound synthesis process.

It is also common, but not mandatory, for each component to provide a *Graphical Interface* (GI) to interact with the physical model, for configuration and control purposes in particular. We consider this option as default, as there are seldom cases where a component will not have even a minimal graphical interface. In our case, the main objective of the graphical interface is to publish the adjustable physical parameters that control the model to the user, who can interfere in the sound synthesis process by modifying these parameters.

The *controller* is an essential part for every component. Its task is to interconnect the graphical interface with the physical model. It also supervises the communication process among the component it belongs to with other components.

Inter-component communication in KTISIVIOS is specified through a messaging protocol. The communication layer of the framework is used to transfer messages between components. Every message signals events. These events are transmitted between components through a broadcasting system based on the Publisher-Subscriber prototype.

KTISIVIOS defines the form, syntax and semantics of these messages. To be more precise, every message must contain a human readable description of the event that signals (e.g. Blow_pressure, Temperature, Pluck). It must also contain timestamps and synchronization details, as well as the numerical values defining the magnitude of the event.

Furthermore, extensibility of messages is of great importance and is supported by KTISIVIOS to allow transmission of messages containing extended information. This protocol draws inspiration from established protocols such as SKINI [18] and OSC [19].

Remotability is another core feature of KTISIVIOS. It enables VMI synthesis with components stored in independent computational systems, connected through a network. VMI applications based on physical modeling

could be demanding in terms of computational resources, due to the simultaneous solution of complicated equation systems. Thus, the distributed operation of the application is extremely important. We have incorporated this functionality in KTISIVIOS using the .Net remoting mechanism.

C. *Product Life Cycle*

The effect use of our framework is based on a particular product development lifecycle that we propose. The fundamental roles and entities in this lifecycle, also shown in Fig. 3, are:

- *Component Developers*: They receive requests for developing VMI components that provide specific functionality and submit them to the Active Repository. Development inside our proposed life cycle is bound by the specifications and guidelines defined by the framework KTISIVIOS and provided by the Active Repository.
- *Integrators*: They receive requests from end users for specific VMI products. Having clarified user requirements, integrators turn to the Repository to look for components that provide the requested functionality. After collecting the necessary components, they use tools provided by the framework to put together the separate components into a complete application.
- *End Users*: They are simple or expert users that request a VMI product from Integrators.
- *Active Repository*: It stores, validates and catalogues implemented components, while providing specifications and guidelines for component development.

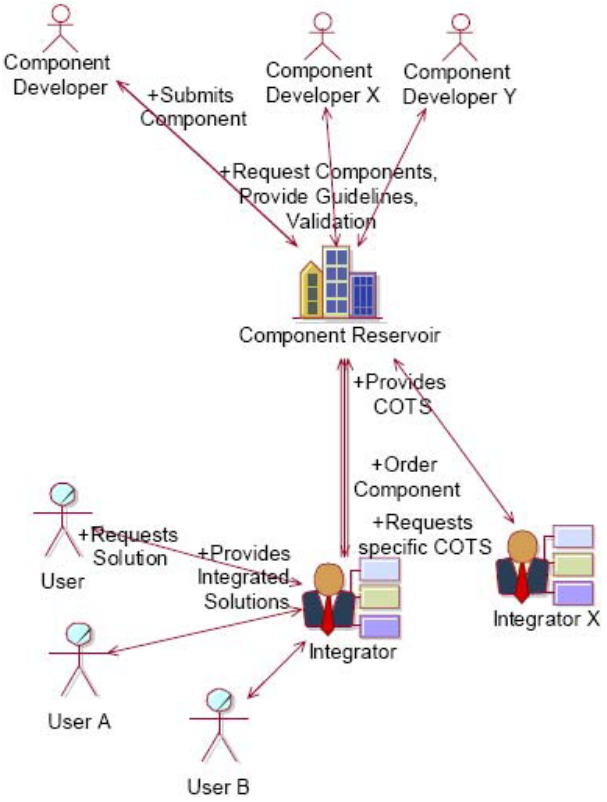

Fig. 3. Roles and product life cycle

Following the proposed lifecycle, components that come from different software vendors and implement different physical modeling approaches for the corresponding instrument parts can incorporate in a reliable way. Thus, a component bank can be created corresponding to a real world instrument parts bank. End users, in collaboration with integrators, select the combination of components that will produce an integrated VMI application that satisfy their needs. The distributed functionality of KTISIVIOS enables the creation of higher order applications using remote components, offering possibilities such as creating Virtual Music Orchestras.

D. *Framework Tools*

KTISIVIOS provides two separate tools for integration and execution of a VMI instrument, namely *Integration Tool* and *Execution Tool*.

The "Integration Tool" provides the integration environment, where components are represented by icons. Integrators connect to the Active Repository, browse through the implemented components and load the appropriate ones. Through the tool's interface, they interconnect the VMI components and arrange their graphical interfaces, formulating step by step the final VMI application. Component connections are directed links, starting from the component that in general requires a service and ending to the component that provides the particular service.

The Graphical User Interface (GUI) of the VMI is also designed through the Integration Tool. The interfaces of each component are arranged by will, and a "snapshot" is taken in order to store the current GUI instance. The stored snapshot will be the GUI of the VMI.

The "Execution Tool" is the final product presented to the end users. It loads the components arranged with the integration tool, as well as the designed GUI, and simulates the corresponding real world instrument.

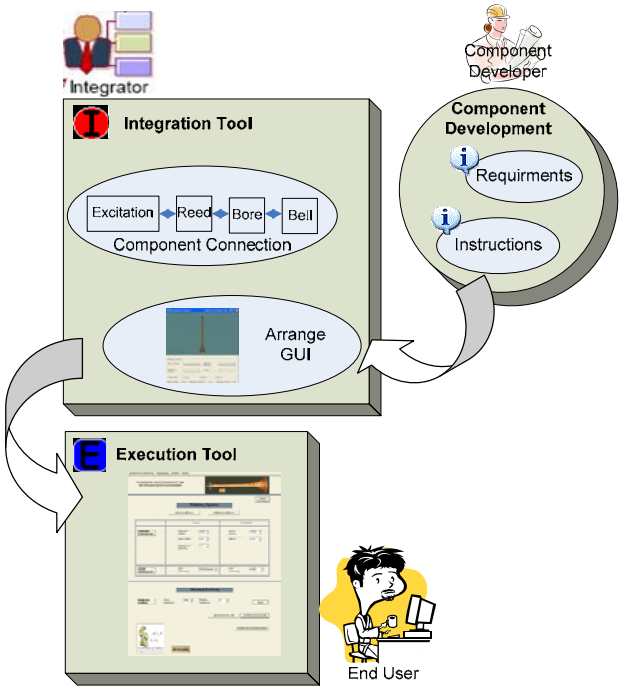

Fig. 4. Framework tools and roles

IV. VIRTUAL ZOURNAS

In this section we demonstrate the implementation of Virtual Zournas, using KTISIVIOS and the proposed life cycle. In the following paragraphs, after giving some information on the zournas, we describe the developed components, their structure and functionality. Secondly we demonstrate the integration of Zournas from these components using the Integration Tool. Finally we experiment with the final product, presenting several use cases.

A. The Zournas

The zournas is a traditional Greek double-reed woodwind instrument, and has a long history. It is also found throughout Europe, North Africa, Middle East, India and China with various names. It resemblances the shawm, the most widespread woodwind double reed instrument of the Middle Ages, which is considered to be the ancestor of the modern oboe [20].

As most woodwind instruments, it consists of two major parts, also shown in Fig. 5: the embouchure and the bore. The embouchure's main part is double reed, which acts as a pressure controlled valve. Through a connector, which is covered with fiber, the embouchure connects to the instrument bore. The bore is conical, like the one of the oboe. The number of tone-holes is usually 7 and there are no keys or other mechanical parts. The bore ends up at a bell, whose flare varies greatly among the instruments found [20].

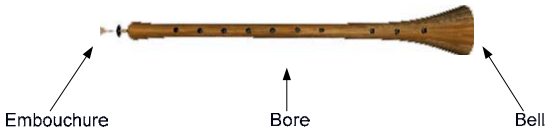

Fig. 5. Image of the zournas

B. Building Components

In order to implement Virtual Zournas, we begin with designing the components that correspond to the functional parts of the real-world zournas. It should be made clear that there can be many approaches for defining the parts that these components describe.

Instrument functional parts correspond to acoustic procedures. Such procedures are by nature modular, in the sense that a single procedure can be described by partial ones. Likewise, an instrument part can be modular. For example, when one defines the "bore", she could consider the "bell" either as part of the bore or as an individual part that connects with it. The "excitation", "oscillating body" and "resonance body" are the most generic, since they are found in all instruments. The excitation mechanism for our case is blowing, the oscillating body is the double reed and the resonance body is the air column inside the bore.

Moreover, there are different ways to describe an acoustic procedure. There are many physical models for the same part, even though each one can have different rate of success. In the same way, there are many approaches regarding a component. A "bore component" can enclose a "bell component" or not, can model the tone-hole behavior or not etc.

We continue by analyzing the components for our VMI. The first three apply to instrument parts, while the later two apply to useful VMI functions.

1) The embouchure component

As its name states, this component corresponds to the embouchure of the zournas. It's most vital parts have to do with the physical model of the embouchure and are the mathematical equations describing double reed's oscillation along with the air's flow characteristics. The acoustical behaviour of double reeds is yet under study. In our approach we use the physical model presented by Almeida et al [21], adapted to measured physical properties of zournas' reeds [20], [22].

In order to represent these equations in a computational system, a discrete time transformation is crucial. This step follows the directives of digital synthesis framework for wind instruments presented by Guillemain et al [23], [24], [25]. We use the same framework for the discretization of the equations describing the other instrument parts and for the design of the overall digital synthesis algorithm.

In TABLE I we present the equations and the corresponding physical variables describing the acoustical process in the embouchure. The model calculates the pressure at the end of the embouchure or the input of the bore when provided with external blowing pressure value.

TABLE I.
EQUATIONS FOR ZOURNAS'S PHYSICAL MODEL

$\frac{d^2y(t)}{dt^2}+\frac{\omega_o}{Q}\cdot\frac{dy(t)}{dt}+\omega_o^2 y(t)=\left[p_r(t)-p_m\right]\cdot\frac{A}{m}$	y: displacement of two reeds ω_o: resonance frequency Q: quality factor p_r: reed pressure p_m: blowing pressure m: mass of the reeds A: reed surface
$S_r(t)=y(t)\cdot w$	w: reed width S_r:reed opening
$q(t)=S_r'(t)\cdot C_r(t)=aS_r(t)\cdot C_r(t)$	q: flow α: Vena Contracta parameter C_r: air stream speed
$C_r(t)=\sqrt{\frac{2}{\rho}\left(p_m-p_r(t)\right)}$	ρ: air's density
$p_b(t)=p_r(t)-\frac{1}{2}\rho\psi\frac{q^2(t)}{S_{ra}^2}$	ψ: Bernoulli parameter S_{ra}:Embouchure's exit surface

Through the component's graphical interface the user can adjust the following parameters:

- Initial reed opening
- Reed width
- Diameter of the embouchure's output

2) The bore component

The bore component implements the physical model of the zournas's bore, including the tone-holes and the bell. The model calculates the bore's impedance, whose expression depends on its geometrical properties and the tone-hole lattice.

The model calculates the external pressure p_{ex} at the exit of the bore, provided with the pressure at the input of the bore (embouchure's output) through the equation:

$$p_{ex} = \frac{d}{dt}\left(p_b^* + q^*\right) \quad (1),$$

where p_b^* and q^* denote the dimensionless pressure and flow variables in the bore. Thus, the embouchure's component output is the input for the bore component. Using Guillemain's framework these equations are transformed in the discrete time domain.

The component's graphical interface publishes to the user the following adjustable parameters:

- Bore diameter
- Bore length
- Bell diameter
- Bell flare

3) The excitation component

The excitation mechanism, blowing for our case, is simulated through this component. The physical model describes the blowing procedure and drives the digital synthesis algorithm for the sound production. The algorithm feeds this value to the embouchure component's input and initiates the sound synthesis procedure.

Even though this component's functionality precedes those of the embouchure and the bore we present it last, because it is natural for the user to fine tune the embouchure's and the bore's parameters, before exciting the instrument via blowing it.

Since the entire sound synthesis algorithm is driven by this component, we choose it to handle the communication between the algorithm's output and the computer's sound system as well. In other words, it is responsible for streaming the physical model's output (acoustic pressure) to the system's soundcard, in order to listen to the sound.

The graphical interface provides the user the option to adjust the blowing pressure and to fire up the sound synthesis procedure by pressing the specified button.

4) The view component

This component does not represent a functional part of zournas, but implements typical functionalities of computational applications like loading and saving configuration files, help, information, and application exit. Thus, a physical model is not implemented.

The most vital part of this component is its graphical interface. Furthermore, it provides two different visual representations of the zournas, a 2-Dimensional and a 3-Dimensional representation.

The 3D-representation of zournas is developed in VRML. The user can examine zournas through the 3D-model and adjust its physical characteristics using the computer's input devices (e.g. mouse). The controller transfers the new values to the other components. The adjustable parameters are:

- Bore length
- Bore diameter
- Bell flare
- Point of view (visual parameter)
- Bore material (visual parameter)

Using the 2D representation, the user can select several possible fingerings, by opening and closing tone-holes, using the computer's mouse. The controller transfers the appropriate fingering values to the bore component, where the tone-hole model is implemented.

5) The execution component

This component implements functionalities for controlling Virtual Zournas and performing with it. To be more precise, it supplies Virtual Zournas with the function to play a melody from a MIDI file or to be controlled by an external MIDI device. MIDI messages are transformed into appropriate values for the parameters of the physical models.

C. Integrating Virtual Zournas

After implementing the aforementioned components, we use the Integration Tool in order to connect the components and design the GUI of the VMI, hence integrating the complete application. Initially, the five components are loaded to the Integration Tool. Afterwards, the communication paths amongst them are specified. This is accomplished using the tool's graphical interface, simply by drawing lines between the components that interchange messages.

Due to the form of the digital sound synthesis algorithm, the excitation component that drives the synthesis procedure connects with the embouchure component and the bore component. The view component is responsible for adjusting the parameters of the embouchure and the bore components. For that reason, it is connected with these components so that the appropriate messages are exchanged.

The execution component connects with the excitation, the embouchure and the bore components, in order to transmit the transformed MIDI messages. At this point, the VMI is almost finished. We arrange the overall graphical interface of the application, by selecting the "arrange GUI" option and place each component's graphical interface in the position we wish to appear in the final application.

Finally, we take a "snapshot" of the arrangement. The snapshot stores not only the designed GUI, but the interconnections among the components as well. In other words, the snapshot stores every action that has taken place in the Integration Tool, for the time being. Now, when the Execution Tool is run, the snapshot is loaded and we have Virtual Zournas on our hands, ready for use.

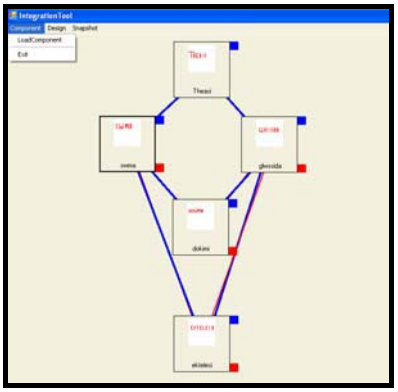

Fig. 6. Components connected using the Integration Tool

D. Using Virtual Zournas

The end user runs the Execution Tool in order to use Virtual Zournas. In the window appearing, one can see the instrument parameter value set, the excitation section, a 2D image of the zournas, function buttons and menus. Let us now examine some use case scenarios that Virtual Zournas offers.

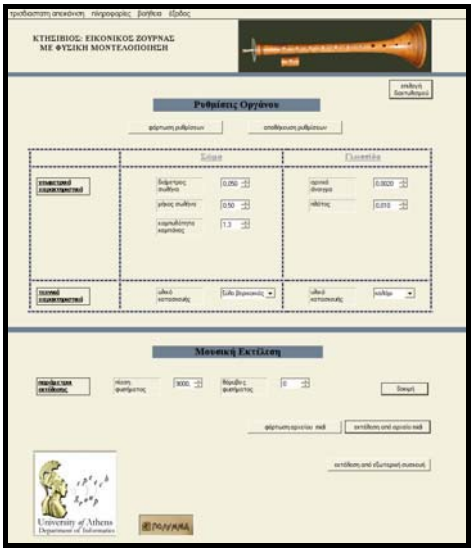

Fig. 7. GUI for the Virtual Zournas

1) Listening to isolated notes

By running the execution tool for the first time, Virtual Zournas loads a default value set for the adjustable embouchure, bore and excitation parameters, described in paragraph IV.B. The user can set the values for these parameters using the computer's input devices. These values must belong into a specific for every parameter value range, since the models cannot produce sound for any given values. The appropriate range, along with a full instruction set, is provided to the user with the help function. If a given value is off range, a warning message appears.

After adjusting the values of the parameters, the user can hear the sound result from the Virtual Zournas he just defined. First, he sets a value for the blowing pressure, in the excitation section. Then, by clicking the "Blow" button, sound is produced for 3 seconds, and can be heard provided a usual computer sound system (soundcard, speaker) is present. The sound corresponds to that produced from a real-world zournas with properties defined by the parameters' values and all tone-holes closed. Of course, it is possible that the values describe a zournas entirely unknown to the real world.

If the user finds the sound result appealing, he can save the parameter value set, in order to load it in later time, without having to re-set the values from scratch. Values are saved locally as an XML formatted file. We define these files as *configuration files*. Blowing pressure value is not saved, since it is considered to be a per

2) Using the 3D replica

By selecting the "3D visualization" menu, the 3D replica of zournas is activated. It is a highly detailed 3D VRML visual model of a real-world zournas. The user manipulates the model, turns it around, flips it over, changes the point of view etc, using the computer's mouse and the slide-bar "Move".

Moreover, she modifies the model's features, described in IV.B.4), using slide-bars, and observes the visual model change accordingly. The values of parameters are shown in the lower part of the window. Apart from affecting the visual model, the values are also sent to the sound model's components. In this way, the user "blows" again and listens to the sound produced from the zournas set up with the 3D interface.

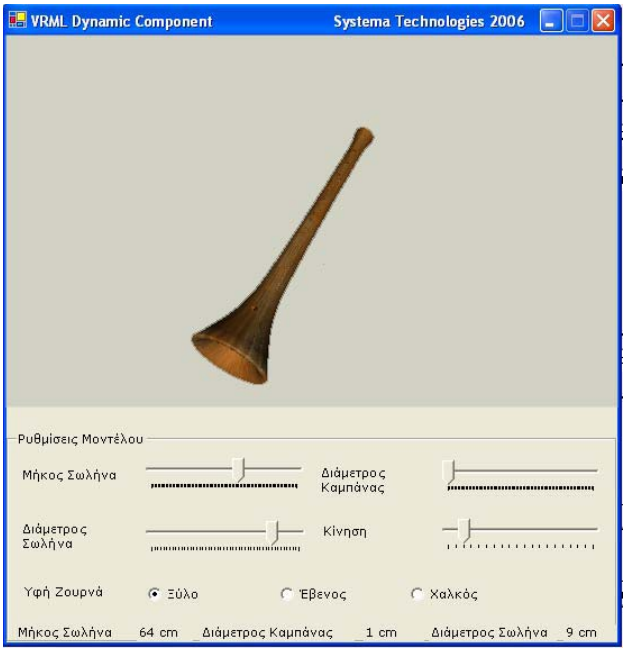

Fig. 8. 3D Visualization of the zournas

3) Choosing fingering

The 2D image of the zournas is located on the upper right part of the main GUI. The user selects a desirable finger by double-clicking over the tone-holes. Open holes appear black, while closed holes appear light brown.

The fingering is confirmed by clicking on "Confirm Fingering" button, and the Virtual Zournas is ready to be blown. The sound result matches that of zournas with the current parameter value set and the confirmed fingering. It is worth mentioning that a fingering cannot be saved and loaded in the way a parameter value set can.

Fig. 9. 2D replica of zournas for choosing fingering

4) Playing a tune with the Virtual Zournas

After experimenting with sounds and fingerings, the user chooses to listen to a whole melody and not just isolated notes played. There are to ways to make Virtual Zournas carry a tune: by loading a MIDI file and by connecting external MIDI devices.

By clicking on "Load MIDI File", the user browses and loads a MIDI file. Frequency and MIDI messages are used by the sound production models, and the melody is played with the sound of zournas. The MIDI file must contain a monophonic melody, since the Virtual Zournas cannot produce more than one note simultaneously.

Perhaps the most interesting case is when the user connects an external MIDI device to control the Virtual Zournas and perform with it, in real-time. By selecting "Connect External Device", a list of the computer's MIDI devices connected to the user's computer shows up and, if the external device is properly installed, it appears on it. By marking it and clicking "Start", the user is ready to perform. Again, frequency and time MIDI messages are used for the sound production sequence. At the end of the performance, the connection is stopped by pressing the "Stop" button.

V. Conclusions and Future Work

Virtual Musical Instruments play a significant role in modern music technology applications. In this paper we presented KTISIVIOS, a component-based framework for developing VMIs based on physical modeling. By following the proposed development life cycle, reduced-cost VMI applications can be provided to a broad range of 'music' users. Ease of customization per user, shorter development times, trivial maintenance and effortless upgradeability of VMI applications are some of the expected benefits of our approach.

Virtual Zournas is the first VMI implemented under the proposed framework. Further research will include improvement of the physical model of zournas, and an improved GUI for Virtual Zournas. Hopefully, many components for various instrument parts will be developed, and the example of Virtual Zournas will be the start point for a series of VMIs designed using KTISIVIOS.

Acknowledgement

This research was funded by European Community Funds and Hellenic National resources under the HERON project of the Research Programme PYTHAGORAS I/ EPEAEK.

References

[1] J. Doble, *Elemental Design Unique Mallet Percussion*, http://www.tidewater.net/~xylojim/

[2] G. Pavlidis., D. Tsiafakis, F. Arnaoutoglou, C. Chamzas, G. Provopoulos, S. Chatzopoulos, "MOMI: A dynamic and internet-based 3D virtual museum of musical instruments", *3-rd International Conference on Museology*, Mytilene, 5-8 June 2006.

[3] S. Goto, T. Suzuki, "The Case Study of Application of Advanced Gesture Interface and Mapping Interface, - Virtual Musical Instrument " Le SuperPolm" and GestureController "BodySuit"", *Proc. of the 2004 Conference on new Interfaces for Musical Expression (NIME04)*, Hamamatsu, Japan, June 3 – 5, 2004.

[4] N. Castagné, C. Cadoz, "The GENESIS environment : physical modeling as a language to be practiced by musicians", *International Conference Acoustics (ICA)*, 2004/04/04, Kyoto, Japan (2004) pp.4 pages

[5] Tassman 4, Available: http://www.applied-acoustics.com/tassman.Htm

[6] V. Välimäki, T. Takala, **"**Virtual musical instruments - natural sound using physical models", *Organised Sound*, Volume 1 , Issue 2 1996, pp. 75 – 86, ISSN:1355-7718.

[7] J. O. Smith III, "A history of ideas leading to virtual acoustic musical instruments", *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2005, 16-19 Oct. 2005, pp. 299 - 306

[8] E. Miranda, M. Wanderley, *New Digital Musical Instruments: Control and Interaction beyond the Keyboard*, A-R Editions, Spring 2006. ISBN 0-89579-585-X

[9] A. Mulder, "Virtual musical instruments: Accessing the sound synthesis universe as a performer", *Proc. of the First Brazilian Symposium on Computer Music*, 1994

[10] J. Rovan, M. Wanderley, " Gestural Controllers : Strategies for Expressive Application", *SEAMUS'98*, Hanover - NH, 1998

[11] *Virtual Instrument Museum:* http://learningobjects.wesleyan.edu/vim/

[12] C. Roads, *The computer music tutorial*, Cambridge, Massachusetts, MIT Press, 1996, p. 265.

[13] R. Rabenstein and L. Trautman, "Digital sound synthesis by physical modelling", *Proc. of Symposium on Image and Signal Processing and Analysis (ISPA'01)*, Paula, Croatia, June 2001.

[14] Reaktor 5, Available: http://www.native-instruments.com/index.php?id=reaktor5_us

[15] Max/MSP,Available: http://www.cycling74.com/products/maxmsp.html

[16] SuperCollider, Available: http://www.audiosynth.com/

[17] D. D'Souza, A. C. Wills, *Objects, Components and Frameworks with UML*, Addison Wesley, Reading, Massachusetts, 1998, p 31.

[18] SKINI: http://ccrma.stanford.edu/software/stk/skini.html

[19] OSC:http://www.cnmat.berkeley.edu/OpenSoundControl/OSC-spec.html

[20] P. Tzevelekos and G. Kouroupetroglou, "Acoustical analysis of woodwind musical instruments for virtual instrument implementation by physical modeling", *Proc. of the Conf. ACOUSTICS 2004*, 27-28 Sept. 2004, pp. 49-60.

[21] A. Almeida, C. Vergez, R. Caussé, X. Rodet "Experimental research on double reed physical properties and its application to sound synthesis", *Proc. of Stockholm Musical Acoustics Conference*, Stockholm 2003.

[22] P. Tzevelekos, T. Perperis, G. Kouroupetroglou, "KTISIVIOS: Virtual Musical Instrument environment, software, documentation", Technical Report, Research Programme *POLYMNIA: Integrted System for Music Tools and Music Portal*, Athens 2006.

[23] Ph. Guillemain, J. Kergomard, Th. Voinier, "Real-Time synthesis models of wind instruments based on physical models", *Proc. of Stockholm Music Acoustic Conference (SMAC)*, pp. 389-393, 2003.

[24] Ph. Guillemain, A digital synthesis model of double-reed wind instruments, *EURASIP Journal on Applied Signal Processing*, 2004:7,pp. 990-1000, 2004.

[25] Ph. Guillemain, J. Kergomard, Th. Voinier, "Real-time synthesis of clarinet-like instruments using digital impedance models", *Journal of the Acoustical Society of America*, pp. 483-494, 2005.