SMS3d: An application for the visualization of SMS data

Bernardo Macías Bécares
Audiovisual Institute, Pompeu Fabra University
Rambla 31, 08002 Barcelona, Spain
bmacias@iua.upf.es http://www.iua.upf.es

Abstract

SMS3d is an application for the three-dimensional visualization of the SMS analysis of a sound, with which the user can control the viewing of the representation with specific controls. This application has been programmed with Visual C++ under Windows using the API OpenGL.

1 Introduction

Two approaches have dominated the representation of musical signals: temporal representations and frequency representations [1].

There are two basic categories for frequency representations: static and time varying [2]. The static representations capture the instantaneous spectrum of a sound and it is projected into a two-dimensional plane where the vertical axis is amplitude and the horizontal on frequency. But, the spectral characteristics of a sound are constantly changing in time and we recover this important property of the sound in the time varying representations, where we see the amplitude and frequency changes in the spectrum. A traditional time-varying representation is the spectrogram. A spectrogram shows frequency in the vertical axis, time in the horizontal axis, and amplitude as gray levels or colors. If we convert the amplitude into a third axis we have a three-dimensional representation.

The SMStools program [3] is a graphical front-end for the SMS (Spectral Modeling Synthesis) analysis/synthesis environment [4]. With it, we can view the sinusoidal plus residual representation with a display similar to a spectrogram. One possible improvement is the three-dimensional representation of the same data points, including a user interaction with the representation.

This paper describes the SMS3d application which features the visualization of a sound analysis made with SMS. In section 2 we describe the different environments for programming 3D graphics. Section 3 describes the implementation and use of the application and section 4 presents some conclusions and possible improvements.

2 3D environments

At the time we started this project we knew of more than 50 APIs (Application Programming Interface) that could have been used for our purpose [5]. Which one do we chose?

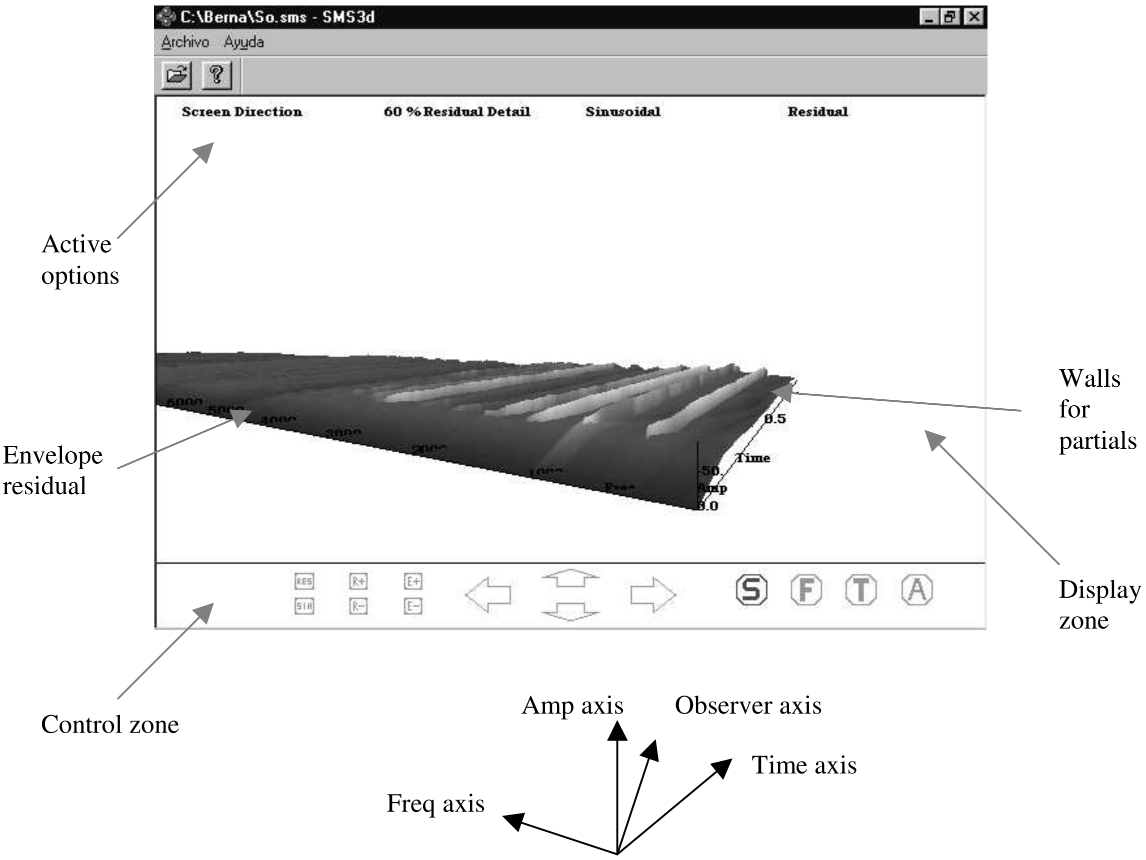
The three applications that we considered were: Silicon Graphics' OpenGL, Apple's QuickDraw 3D, and Microsoft's Direct3D.

For cross-platform work, OpenGL (available for UNIX, Windows NT and 95, and Mac OS) is clearly the choice, especially for developing technical applications. Its client/server architecture enables desktop systems to off-load graphics processing to a server. QuickDraw 3D (QD3D) also has an advantage here, running on both Power Macs and Window (NT and 95).

OpenGL, however, does not handle some high-level functions (ex. File formats). QD3D and Direct3D both remove that particular headache from your development list. QD3D also has hardware abstraction, plus ready-made objects and built-in editing support.

If you program for the Mac as well as Windows, you want to develop for QD3D. If you want to take full advantage of PC hardware (as games might), both Direct3D and Apple's RAVE (Rendering Acceleration Virtual Engine, the device-independent interface that QD3D uses) offer solutions.

You can call OpenGL [6] from a variety of languages, including C/C++, FORTRAN, Ada, and Java. It support both an *immediate* mode and a *retained* mode for graphics operations. In the immediate mode, an application sends graphic commands to OpenGL, which promptly executes them. In the retained mode, sequences of graphic commands are stored in data structures known as *display lists*. You only have to reference its display list to display an object.



OpenGL supports a wide range of graphics environments that cover the spectrum in terms of cost and performance. On the low end, it provides software-only rendering for desktop PCs. But it can also communicate directly with high-end workstations equipped with visualization hardware that can draw 110 millions polygons per second. With its industrial-strength graphics primitives, it is ideal for CAD and architectural design programs.

3 SMS3d

SMS3d is an application developed in Visual C++ [5] under Windows and uses the API OpenGL for three-dimensional graphics. From an input SMS file it displays the sinusoidal and residual information that it contains.

3.1 Graphical display

Each partial is represented as a time-varying amplitude and frequency envelope. Points of each frame are joined and trajectories are drawn like “walls”. We display the envelope of the magnitude of the residual, and its change in time, as a three dimensional surface. This surface is filled with rectangles that vary in color without any empty space.

3.2 Workspace

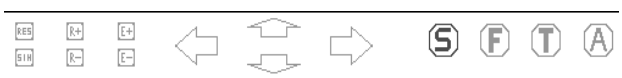
A unique workspace shows the three-dimensional representation of the analyzed sound. This workspace is divided in two parts; one for the graphic representation, the other for the movement and representation of the visualization controls.

Allowed operations for the representation are:

⇒ Increase or decrease of the axis scale for a better visualization.

⇒ Increase or decrease of the residual resolution. This operation has been implemented because when representing all the residual data, the application suffered and excessive slow-down. That’s why it was decided to allow the user to decrease the number of data points represented via the resolution control.

⇒ Choose whether to display sinusoids, residual or both.



Control zone

From the control area the user can access all the operations existing in the application. Partials (SIN) and residual (RES) can be showed, residual resolution can be increased or decreased (R+ and R-) and selected axis scale increased or decreased (E+ and E-). With the control arrows in the middle of the control area, the user moves and rotates the display. The up and down arrows allows the movement in positive or negative direction of an axis. In case the S axis is selected (observer direction), the representation is approached or moved away. The left and right arrows make the representation rotate anti-clockwise and clockwise respectively in the axis direction. If the S axis is the selected axis, the result will be in "we rotating". The different axis for the movement or scale are shown in the right.

3.3 Software modularization

For the development of the Visual C++ code, the application was divided into one class for each of the main processes involved.

The following classes were developed:

- *CRepresentation3D*
  - Controls the partials and residual display.
  - Initializes and calculates the colors of the representation.
  - Renders the represents, axis and control area.
  - Supervises the control zone.
  - Initializes the display zone.
  - Calculates the representation initial position.
  - Controls the window size changes.

- *CTrans3D*
  - Increases and decreases the axis scale.
  - Realizes the movements through the representation.

- *CMath3D*
  - Calculates the plane normals.
  - Calculates the matrix multiplication.

3 Conclusion

The application gives the user a more expressive vision of a sound than the current graphical application used for SMS. In order to watch different aspects and details of the sound the user can manipulate the representation interactively.

The selection of OpenGL allows to do each three-dimensional draw on a Windows system. It has a very high flexibility because it is only necessary to specify the figure vertex and it will be displayed automatically. Also, it has a wide range of effects and many transformations can be easily done. A disadvantage is the necessity of representing simple objects (e. g. the arrows of the application control system). Each single graphical object has to be completely defined. Also it is not easy to insert text into the application (e.g. the text in the control area has been drawn using bitmaps in order to make it different from the axis text).

This application can be expanded in many new directions. Simple extensions would be to incorporate the edition of the sound being represented, to add the capability of representing more than one sound at a time, to investigate the use of an accelerator card, and finally to incorporate this application into the current SMS graphical front end.

4 Acknowledgements

I would like to thank all the members of the Music Technology Group of the Audiovisual Institute for their support in this project.

References

[1] G. de Poli, A. Piccialli and C. Roads. [eds.]. *Representations of Musical Signals*. Cambridge, Massachussetts: The MIT Press, 1991.

[2] C. Roads. *The Computer Music Tutorial*. Cambridge, Massachussets: The MIT Press, 1996.

[3] Music Technology Group. *SMS Homepage*. http://www.iua.upf.es/~sms/. IUA-UPF.

[4] X. Serra. "Musical Sound Modeling with Sinusoids plus Noise". G. D. Poli and others (eds.), *Musical Signal Processing*, Swets & Zeitlinger Publishers, 1997.

[5] D. J. Kruglinski. *Programación avanzada con Visual C++*. McGraw-Hill, 1996.

[6] T. Thompson. *"Must-See 3-D Engines" Online*. http://www.byte.com/art/9606/sec11/art4.htm 21 may 1998.

[7] J. Neider, T. Davis and M. Woo. *OpenGL Programming Guide. The Official Guide to Learning OpenGL, Release 1 (Red Book).* Addison-Wesley, 1993.