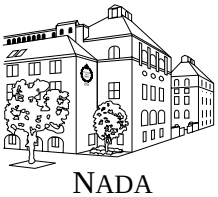

NADA
Numerisk analys och datalogi
Kungl Tekniska Högskolan
100 44 STOCKHOLM

Department of Numerical Analysis
and Computer Science
Royal Institute of Technology
SE-100 44 Stockholm, SWEDEN

# *Auditory Display and the VTK Sonification Toolkit*

*Master's thesis*

Sarah Amandusson

TRITA-NA-Eyynn

Master's Thesis in Computer Science (20 credits)
at the School of Engineering Physics,
Royal Institute of Technology, August 2003
Supervisor at Nada was Kai-Mikael Jää-Aro
Examiner was Yngve Sundblad

# Abstract

The main task of my master's thesis was to find out more about sonification, its uses and whether it could be interesting to develop this area of research at KTH. With this background information I got the task of developing a sonification tool for one of the main visualization toolkits used at KTH.

Auditory Display (AD) is explained and some light is shed over possible uses along with problems and difficulties of this area of research. Some software for sonification is being described along with comments on usability, mapping etc. The tested software include Listen and JASS. Volumetric data sets are explained and difficulties with sonification of these kinds of data along with possible solutions.

My sonification toolkit is made for files written in the Visualization Toolkit (VTK) format. The program is thoroughly described and links for download of the toolkit and midi files are available.

**Tumregler för sonifiering och introduktion till "VTK Sonification Toolkit"**

Examensarbete

# Sammanfattning

Min uppgift var att ta reda på mer om fördelar och nackdelar med sonifiering samt att studera olika sonifieringsprogram för att se om detta kan vara ett intressant område för vidare forskning på KTH.

Sonifiering beskrivs och fördelar, nackdelar och problem gås igenom tillsammans med en sammanställning av forskning inom området i nuläget. Några, icke-kommersiella, program för sonifiering testas och beskrivs. Listen och JASS testades. Inga kommersiella program har testats men jag tar upp några exempel.

Mitt verktyg för sonifiering av datafiler gjorda för the Visualization Toolkit (VTK) beskrivs i detalj och länkar för nerladdning av programmet och exempelfiler bifogas.

# Contents

# Chapter 1

# Introduction

Sonification is the use of nonspeech audio to convey information [23]. More specifically, it is the mapping of data relations to sound in order to enable or enhance the finding of patterns and relationships in that data [12]. Some very successful examples include sonar, alarm systems and Morse code. Sonification is closely related to audification—the direct playback of data samples, and auralization (also known as audiation)—the auditory representation of data, i.e. virtual acoustics (3-D sound) [21].

Sonification, especially in combination with some other medium, has been proven to be very useful in many different areas. In order to increase the use of sonification methods there are three different things that need taking care of according to an National Science Foundation (NSF) report written in 1999 [23]:

1. Sonification tools for research and application need to be developed
2. Psychological research. Perception and cognition of sound need to be understood in a better way.
3. Guidelines for sonification design and application are needed

These criteria are the same as in the first NSF sonification report written over ten years ago but are still of immediate interest. Traditionally the use of sonification has been very limited but the research area grew in the 1980s and 1990s and looking at the number of papers from the past few years the area seems to have stagnated a bit and there are not that many new findings on "pure" sonification. This might be because it has proven difficult to get research funding for sonification only. However, there is a lot of research going on in related areas such as virtual environments, computer games and developing aids for the blind. Along with a description of auditory display in general, I will analyse the problems above and their possible solutions.

My own sonification tool, to be used with the Visualization Toolkit (VTK) [38], will also be described thoroughly.

# Part I

# Auditory Display

# Chapter 2

# Auditory Display

## 2.1 Preliminaries

In everyday life people communicate using all their senses but in front of a computer communication is restricted almost solely to the visual channel. By using different forms of input and output such as haptic devices and sound we may be able to make the interaction between people and computers more natural and efficient.

Auditory Display (AD) is the mapping of values in some data space onto some parameters in acoustic space [27]. It includes all types of sound, but I will focus on non-speech sound. This far sonification research has centered on tasks where the eyes are busy—alarm, status displaying, games, aids for the blind, etc. The goal of an AD is to reveal some unknown features, or help clarifying important relations in the data sets together with other techniques for displaying data, mainly visualization.

### 2.1.1 Visual and Auditory Display

Due to most direct manipulation tasks on a computer being graphical the audio channel is usually free and available as a channel for notification. Using audio has many advantages, the most important being that people can process audio information while simultaneously working on something completely different since sound does not take up screen space. We use this ability all the time without thinking of it, we can listen to radio and read simultaneously, we can choose to listen selectively and switch attention from one sound to another [11] (known as the "Cocktail party effect" because of the possibility to attend to one conversation in the midst of many others).

Audio as a displaying technique is suitable for presenting high-dimensional data without creating information overload for users [23] and also for noting correlations and repetitions [9]. If we listen to a piece of music, we can concentrate on one instrument or, if we want to, appreciate the composite sound of all instruments. This makes the auditory channel a powerful tool in data comprehension but it also takes some training in how to interpret the sounds.

In the 1980s Sarah Bly did a series of studies on classification of nonordered multivariate data sets using visualization, sonification and both simultaneously. She showed that for categorizing data AD and visualization were equally fast but using audio and visual simultaneously was superior. The combination of visual images and sounds provides an extremely powerful tool for uncovering complicated structures. Sometimes the sounds reveal features that are hidden in the visual images and at other times the visual images show features that are difficult to detect using sound. In order to convince people that the total effect will gain from a combination of visualization and sonifiation, sonification must be used more frequently and better tools must be created.

When both methods are used simultaneously we have to decide what shall be communicated through sound and what should be visualized. When that problem is solved another difficulty occurs: If an audio display is preferable—when is it best to use speech and when would an alternative sound be more suitable? Some rules will be given in section 3.

### 2.1.2 Music and Sonification

If an auditory computer interface is overly simple, intrusive or unpleasant it is likely to be turned off even if it is useful. To prevent this from happening the AD designer can use the way a musical composer uses sound to convey emotion in order to make a nicer and better sound [21]. The study of how music is perceived and what feelings it arises also applies to sonification issues. A listener can distinguish one musical piece from others (fine ears distinguish late Mozart from early Beethoven etc.). This effect comes from self-similarity, which is characteristic of music design and perception.

One problem in making a "musical sonification" is that the use of instrument timbres or musical scales may convey unintended cultural and historical meaning and should only be used when the symbolic meaning doesn't contradict or confuse the original meaning of the sonification [37].

### 2.1.3 Examples of Successful Applications using AD

There are numerous applications where AD has been used with good results but to give a feeling of what it might be used for, some examples are given below.

**Sonar** determines the nature of underwater entities. It works by sending a sound signal and picking up/interpreting the answer, which makes it possible to measure distances and localize targets etc.

**Language-like coding** includes Morse code and military signaling (trumpets, drums etc.).

**Warning systems** on aircraft, cars, trains, in buildings, etc. in order to inform people that something unusual has occured.

## 2.2 Sound and Perception

### 2.2.1 Human Hearing

Sound waves are air pressure oscillations caused by the propagation of an initial deformation or displacement of the air [37] and are closely related to some kind of action (if you hear a car close by, you try to get out of the way). Since we have two ears we are able to decide what direction a sound is coming from, which helps us in many everyday tasks. Our hearing is great when it comes to selecting one important sound from a large amount of closely related sounds.

The way sounds are perceived is individual and the acoustic signals will differ due to interactions with the environment. There are *internal differences* (sources off to one side arrive sooner at the near ear) and *internal intensity differences* (sources off to one side are louder at the near ear due to head-shadowing) which also changes the way the sound is perceived.

We can hear sounds as short as 25–50 ms within the span of 20 Hz–20 kHz [27]. Our ear has, like the eye, a very high power of discrimination between periodic and aperiodic events and can detect small changes in the frequency of continuous signals which makes AD good for understanding fast-changing data [23].

### 2.2.2 Auditory Perception

According to Williams [43] *sensation* refers to immediate and basic experiences generated by isolated, simple stimuli and *perception* involves the interpretation of those sensations, giving them meaning and organization. *Synthetic perception* is when the information presented is interpreted as generally as possible (hearing a room full of voices or listening to the overall effect of a piece of music) while *analytic perception* concerns the case of using the information to identify the components of the scene to finer levels (listening to a particular voice in a crowded room) [43].

A thorough understanding of the perception of complex sounds is essential to the development of ADs. Some research in how we perceive music, speech and meaningless sounds can be found. However, usually the sounds we hear tell us something about the environment—they are not meaningless. The perceptual process by which the listener separates out the information from an acoustic signal into individual meaningful sounds is called *auditory grouping*.

Gestalt psychology is the study of the human propensity to recognize patterns and configurations which appear in the environment and it proposes that the ability to perceive form is inborn. Once features are discovered in a complex sound the analyst takes some action, either making a decision based upon the perception or perhaps turning the relevant section of data over to mathematical analysis.

According to Williams [43] and Kramer [21] some processes identified in the context of perception are:

**Similarity** Objects sharing the same attributes are perceived as related. Example: Many people clapping their hands at the same time produces a louder sound

than one person clapping.

**Proximity** The closer two things are the more likely they are to be perceived to belong together. Example: If the frequency difference between two tones is small it is more difficult to perceive them as two independent sound sources.

**Good continuation** Smooth transitions in components from one state to another are perceived as related. Example: Tones separated by noise are more likely to be heard as continuous if the start frequency of the second tone matches the end frequency of the first tone.

**Habit/familiarity** Sounds heard before will be assigned the same meaning when they occur again.

**Belongingness** A component can normally form part of only one disjunctive object at a time and the way it is percepted is relative to the rest of the figure to which it belongs. Example: When the perception related to a particular data attribute is hidden or distorted by its context.

**Common fate** Components which experience the same kinds of changes at the same time are perceived as related.

**Closure** Incomplete figures tend to be completed. Example: The listener tends to percept the sonification as more regular and complete than it really is, thus disguising subtle variations in the data.

**Stability** One interpretation will remain fixed throughout slowly changing parameters until the original interpretation is no longer appropriate.

**Transparency** Components from multiple sources are convolved into complex acoustic waves rather than hiding each other.

It can be seen that changes in the physical parameters of the auditory signal may not lead to corresponding changes in its associated perceptual attributes, which means that the psychophysical interpretation of sound is not linear—some sounds "over glow" others.

### 2.2.3 Drawbacks with the Use of Sound

Most conceptual problems in scientific sonification are related to finding suitable mappings between the space of data and the space of sounds but there are also some more practical difficulties with auditory interfaces [19]. An auditory interface in a speaking environment will very likely cause annoyance unless headphones are used but with headphones on it is more difficult to interact with the environment.

Another major difficulty is that it is almost impossible to get the exact value of the data, since most people only notice whether a sound goes up or down.

A sound might evoke a reference to memorable sounds from other sounds or music and hence may contain unintended messages which the AD designer can not decide over [2].

Another difficulty is to define spatial dependence, and this far it is most often defined using the edges of the object.

### 2.2.4 Studies in Auditory Perception

#### Earcons and Auditory Icons

A lot of the research in auditory perception has been made using auditory icons and earcons. Auditory icons use "real world" sounds to convey messages [5] while earcons are abstract, synthetic tones that can be used in structured combinations to create sound messages to represent parts of an interface [10]. Brewster et al. [10] have shown that earcons with their tones, or sequences of tones, for building messages were better than unstructured bursts of sound and that musical timbres were more effective than simple tones.

It was shown that if musical sounds are used for earcons then musically untrained people would not be at a disadvantage to musicians. It is realistic to believe that more than one earcon might be active at the same time and if so timbre, rhythm and pitch must be clearly distinguishable for separation.

In aircraft simulator tests mentioned in Kramer [21] it has been shown that a maximum of 10 simultaneous alarms can be separated, but only 5–7 can be learnt quickly—after that it takes longer to learn each new signal. It was also noted that warnings with similar change in time are easily confused even if the the sound is different and often speech warnings are preferable. However, it should be noted that pilots tend to turn auditory alarms off if there are too many or if they are disturbing.

#### Interfaces for the Blind

A graphical user interface is inappropriate or unusable when the task requires that the user's visual attention is somewhere besides the computer screen or when the user is blind or visually impaired. In designing auditory interfaces it should be noted that speech has a low information transmission rate for continuously changing variables relative to the bandwidth of the human auditory system.

One of the most widespread myths about blindness is that blind people have a greater ability to perceive and process the auditory information, i.e. better hearing. Winberg [45] claims that Lowenfeld showed in 1980 that there are no studies that support this notion and that it is important to make the distinction between better hearing on a perceptual level and better learned ability to use auditory information.

#### Sound in Addition to Visual Displays in Education

One study made by Bonebright et al. [7] consisted of matching visual and sonified graphs for bivariate and multivariate data sets. The subjects matched one sound

with one of four different graphs. They got mainly good results but when the graph was diffuse and lacked a definite shape the results were bad (although it was still more than 50% correct answers). The results also showed that the matching capabilities had nothing to do with musicality.

### Classification of Objects by Sound

Rocchesso [36] wanted to understand the acoustic features that convey a sense of shape to the listener and made a study using a spherical and a rectangular resonator. Two membranes of different geometries can exhibit the same eigenfrequencies and an acoustic 3-D resonator is almost a linear system and is described by its impulse response or the frequency response. Since we normally use other senses to get shape information, training was necessary.

The main problem in assessing the capabilities of humans in distinguishing shapes from acoustic signatures proved to be the dominance of pitch as a perceived feature, since it is usually associated with size. The classification was better than random choice, and the results were better for large volumes (random choice for volumes with a diameter less than 50 cm).

# Chapter 3

# Heuristics of Sonification

It is a significant challenge to determine how to map data to an audio representation. An ongoing issue in data exploration is how to represent the data to find structure and patterns. The auditory display of scientific data requires consideration of the task at hand, understanding of data characteristics and knowledge of psychoacoustics [3]. There is no applied theory on how to design a successful sonification yet but as sonification theory emerges it will prevent researchers from repeating the errors of prior efforts. Until we understand more about what makes a sonification successful, the field will rely mainly on trial-and-error design.

## 3.1 Auditory Characteristics for Displays

The design challenge is to display the rate of change of the system such that local detail does not prohibit the perception of larger dynamical structures [26]. Some sound attributes available in sonification are timbre, rhythm, pitch, register and dynamics but there are also temporal combinations like sequential, combined or concurrent sounds. We also have to consider what type of data we will sonify, what their numerical properties are, how many dimensions should be handled, if coordination with graphics is necessary or if there is any interaction from the user involved? [5]

In his PhD thesis [4], Stephen Barrass looks on the advantages of a case-based design model called EarBenders. This method would give a natural connection to the data represented and help the resulting sound from being unpleasant. The case-based method relies on a rich source of examples to be effective, and Barrass developed a database with stories about everyday listening experiences, called EarBenders. The goal was to create an effective design technique that is quicker, easier and has the advantage that users will already be familiar with the borrowed elements.

Just as some color models cannot represent every possible color, a sound model may not include every possible sound. The use of a sound model can help standardize the capabilities of sound display equipment [44].

Barrass also suggested that using principles developed by graphic designers, integrated with psychoacoustic observations may provide the systematic approach that has been called for. These principles are called the *Hearsay principles*. An alternative representation of the Hearsay principles is the Information-Sound Space (ISS), a 3-D spatial organisation of auditory relations that bridges the gap from theory to practice by changing the way a designer can think about and manipulate relations between sounds. ISS is based on the HSL (Hue, Saturation, Lightness) colour space which has proven very helpful for understanding and specifying colour relations in information displays.

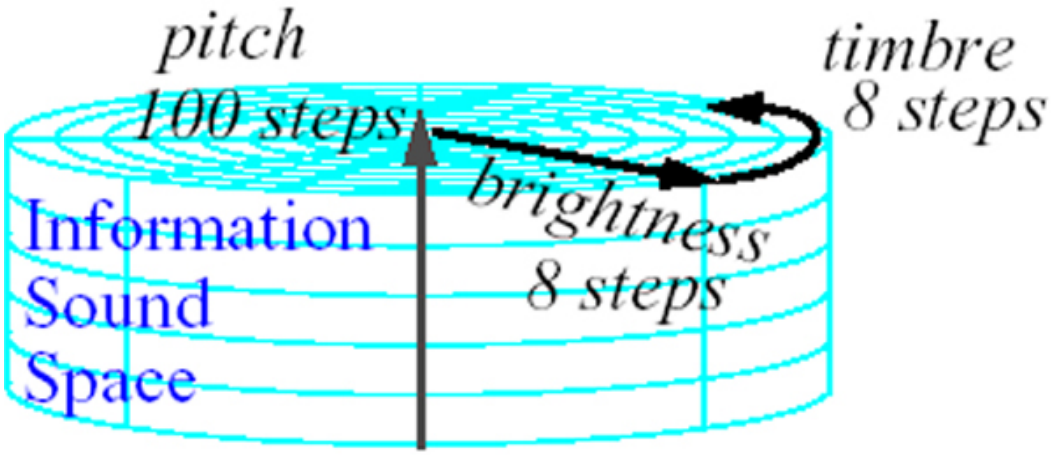


**Figure 3.1.** The Information-Sound Space (ISS) by Barrass [4]

The ISS is a cylindrical polar organization that has a cyclic dimension of 8 categories, a radial of 8 equal, ordered differences, and a vertical axis of 100 equal, ordered differences. The ISS moulds the Hearsay auditory principles into a form that allows a designer to think in terms of spatial relations such as lines and planes, rather than rules. The ISS is structured so that auditory relations can represent a general range of information relations. Unfortunately work on the ISS and the program it was used together with (Personify), ended in 1997.

## 3.2 Design Guidelines

A discrete sound consists of a number of components; frequency, amplitude, location etc.

Some guidelines on how to start designing ADs are summarized below.

**Pitch** is the changing of tones in a musical piece—what makes up the most important part of the melody. Since only a few people have the ability to recognize a separate pitch it shouldn't be used on its own unless there are very big differences between the pitches used. Suggested ranges are from one octave below

middle C (125–150 Hz) to four octaves above middle C (5 kHz) [10]. The relationship between frequency and pitch is logarithmic (in a one-octave interval, the frequency of the highest note is exactly twice the frequency of the lowest note).

**Timbre** is the character of a sound that distinguishes it from other sounds. By using musical instrument timbres the listener will be familiar with the sound and hence be better at recognizing it among other sounds. When using many sounds simultaneously they should be easy to tell apart (brass and organ instead of brass 1 and brass 2) [10].

**Intensity** is the "loudness" of the sound and should be 10–20 dB above threshold. By keeping all sounds within a close range no sound will be lost when the user changes the volume of the system [10]. Intensity is measured in decibels. Doubling the amplitude of a sound increases the loudness by 3 dB and is perceived as only a slight increase. To double the perceived loudness of a sound, the strength of the sound must be increased by 10 dB [44].

**Rhythm** is the timing of notes that are played. This creates the uniqueness of each piece and gives each piece its own character. Sounds with the same rhythm are likely to be confused, even if the timbres are completely different. This can be put right by having different numbers of notes in each rhythm [2].

**Polyphony** usually refers to the number of instrument sounds that a sound module, sampler, synth or midi device can play back at any given time. It usually describes the presence of multiple simultaneous sounds [2].

**Ambiguity** means that a sound can be interpreted in different ways. In sonification it is important to make sure that we get the intended interpretation. Sometimes ambiguity can be valuable in auditory display when it is the intended message. An example of a situation when ambiguity causes negative effects is the "emergent percept" problem that occurs when two simular (but not identical) notes are played simultanously and create a pulsating sound.

**Metaphorical** associations can maximize comprehensibility but since associations are usually subjective it is important to have an artful balance of the associated effects when designing a display [22].

Some more things to keep in mind are that attention is drawn more to some variables than others, i.e. a change in "master pitch" draws a listener's attention which might make other changes more difficult to notice, a display can be better by scaling (i.e. scale one octave to seven octaves), multiple mappings can prove useful (i.e. routing of input data to more than one auditory variable) and finally—try a variety of mappings to emphasize different parts of the display.

Chaotic and unmusical sounds can be hard to listen to. One sound can mask another (just as in graphical displays one object can mask another). If the duration

of a note is too short, we may not be able to perceive its pitch. Sounds interacting with each other may create unintended and confusing effects. Properties of sounds may also interact, since high pitches appear louder than low pitches the use of volume with pitch in a multivariate mapping should be avoided [44].

### 3.2.1 Parameter Nesting and Overlap

Parameter nesting is the use of basic sonic variables on different time scales simultaneously [22]. Several different types of nesting can be considered. Loudness nesting concerns speed, duration, envelope (instantaneous loudness), cluster speed and master loudness while pitch nesting includes master pitch (the overall pitch area), speed, wave shape, degree of quantization of pitch changes etc. Brightness nesting appears when comparing two different instruments, say a flute and a horn, and even though they are playing the same tune it sounds different due to the different frequency content of the sounds.

Parameter overlap occurs when the same audible variable is used (on different time scales) and creates a loss of clarity in one or two nested parameters. Related variables can be profitably mapped to overlapping parameters.

# Part II

# Software and Sound Formats

# Chapter 4

# Software and Sound Formats

## 4.1 Preliminaries

Although sound offers another medium for the representation of complex data, its use in scientific computing has received little attention, mostly because of the lack of suitable software.

The goal of a good sonification tool should include access to low-level signal processing functions to nonexperts, sound synthesis (creating, replicating and transforming) and modeling of the physical properties of a sound-producing object [23]. The following chapters will give a short introduction to different concepts when working with sound on a computer along with some programs that can be used for sonification.

## 4.2 MIDI, General MIDI and Transform Coders

MIDI (Music Instrumentation Digital Interface) was created when musicians discovered a need to control their electronic musical instruments remotely or automatically, without being tied to one particular manufacturer's product or one particular type of instrument. MIDI is a set of commands that electronic devices digitally pass between their MIDI jacks to tell each other what actions to perform.

There are two main advantages of MIDI—it is easy to edit/manipulate and it is a compact form of data [28]. MIDI devices transmit/receive max 31.25 kbits/s through each port, the rest is buffered up. For sonification data that bit rate is subject to unacceptable time delay since there will be a problem with information loss if the sonification data has a higher resolution than 7 bits [21].

On a MIDI sound module the instrumental sounds are made up by patches. Once MIDI became a widely used interface the need to define a standard set of patches grew. Defined patch numbers would make synthesizers and sound cards more compatible. Hence it was decided that patch number one should be the sound of an acoustic grand piano, on all sound modules. The standard includes 128 patches in a specific order and is called *General MIDI* (GM). A GM sound module should

be able to play MIDI events on all 16 channels simultanously (with a different GM Patch sounding for each channel).

There are many formats that can handle higher resolutions while still being a compact form of data. *Transform coders* use information about human perception to selectively quantize regions of the frequency spectrum and "hide" the quantization noise in places where the human ear will not detect it. Transform coding is used in multimedia standard audio formats such as Dolby's AC-2 and AC-3 and MPEG. The MP3 format (audio layer 3 of the MPEG-2 specification) compresses 176 kbytes/s of CD quality digital audio into 128 kbits/s and is in most situations good enough for sonifications [12].

## 4.3 CSound

Csound is a programming language designed and optimized for sound rendering and signal processing and it is the most famous sound synthesis language. It was originally created by Barry Vercoe in 1985 but has recently grown to a huge open source project (download from http://www.csounds.com/). Csound supports audio and MIDI and later versions also support video and 3D graphics. There are an increasing number of graphical user interfaces for the language, but as with most programming languages a word processor is sufficient. When using Csound, two text files are created — one file containing the instruments and one file containing the notes.

In Csound there are no limitations on how many oscillators, filters and combinations there are which is usually the case in commercial hardware synthesizers. It is possible to interconnect the modules, and change the parameters which control them [8]. The complexity of the patches is supposed to be limited by knowledge, interest, and need, but never by the language itself. As far as I know it is very powerful but with the drawback that it is hard to learn and has some difficulties in communication with other programs.

## 4.4 Listen

Listen was created by Catherine M. Wilson in order to encourage the incorporation of sonification into the research environment [44]. She wanted to create a toolkit to be used in exploring data of any type that could easily be incorporated into an existing visualization system. It should also be general and flexible, modifiable, extensible, highly interactive, perform essential routine tasks automatically, easy to install and intuitive to use. The result, Listen, is a data sonification system that allows mapping of data to sound parameters such as timbre, pitch, location, volume and duration [32].

The Listen toolkit consists of four different programs to provide incremental functionality. All versions are made for use with Silicon Graphics machines. Listen 1 is command-line based and uses the internal audio chip. Listen 2 has a graphical

user interface and more complex mappings than Listen 1 but also uses the internal audio chip. Listen 3 also has a graphical user interface but uses a MIDI device and the mapping possibilities are more complex. Listen 4 is a module version of Listen 3 designed to be easily incorporated into an existing visualization program.

According to Wilson a drawback with Listen was that since sounds generated are non-musical, they can be fatiguing when exploring large data sets over extended periods of time. *MUSE* (MUsical Sonification Environment) was created by Lodha et al. in order to create a sonification system a bit less tiring [25]. MUSE maps scientific data to musical sounds; pitch (melody), rhythm, tempo, volume, timbre and harmony.

I had some difficulties when running Listen, most of them due to the time that has passed since Listen was made. I have not found any updated versions of Listen and the graphical packages had changed since the original version. I also had great difficulties trying to get MIDI to work on the SGI O2 I was using.

## 4.5 Java Audio Synthesis System (JASS)

The creators of JASS, Kees van den Doel and Dinesh K. Pai, wanted to make something that was suitable for creating Foley sounds (sound effects added to the film during post production) in an interactive environment with real-time user interactions (computer games, simulations, etc.). The environment is based on a foundation structure consisting of a small number of Java interfaces and abstract classes, and a potentially unlimited number of unit generators. The audio software is implemented using the JavaSound API, which unfortunately requires large buffers for real-time synthesis.

The current distribution of the JASS system contains a set of unit generators for reading and playing audio files, varying speeds and volumes, mixers, various filters, and unit generators for rendering and capturing audio. Their unit generators are connected into filter-graphs, or "patches", equivalent to timbre trees [41]. The Jass SDK and documentation can be downloaded from http://www.cs.ubc.ca/~kvdoel/jass/jass.html.

## 4.6 Other Programs

There are many other tools that are being developed and used, among those are:

**Digital Instrument for Additive Sound Synthesis (DIASS)** was developed by Hans Kaper and Sever Tipei and has the capability to control the loudness of a sound and synthesize sounds that are perceived by the listener as being "equally loud". The perception of loudness is a subjective experience and the loudness routines in DIASS implement relevant results of psychoacoustics research in software [18]. I have not been able to find where to get it or whether it is still being developed.

**Pure Data (PD)** is a real-time, free of charge, software system for live musical and multimedia performances. It is a redesign of an earlier system, still in commercial development, MAX/MSP [33]. It is easy to communicate with from other tools using TCP/UDP and has a visual programming interface. PD is open source software originally developed by Miller Smith Puckette at IRCAM. It is free for any use and can be downloaded at http://www.pure-data.org/.

**Vanilla sound server (VSS)** has been in developement by NCSA's Audio Group since 1993 and is designed to generate synchronized sound in real time for interactive computing environments. The key elements of VSS development were sound production on a computer without a special sound card, real-time interaction, and synchronization to the computer graphics display. It was designed to be a tool for scientists to rely on no matter what platform they used [29]. Software, tutorials and reference manual can be downloaded from http://www.isl.uiuc.edu/software/software.html.

**Virtual Audio Server (VAS)** is a toolkit designed for exploring problems in the creation of realistic virtual sonic environments [15]. It was developed at the Naval Research Laboratory by H. Fouad, J. A. Ballas and D. Brock. The primary thought when creating VAS was the extensibility, which provides the flexibility necessary to support research development. It runs four independent subsystems and offers support for different localization techniques (vector base amplitude panning, HRTFs, intensity panning). VAS can theoretically support a large number of concurrent sounds but due to current limitations in SGI's Audio Library the limit is 16. The VAS software and a beginner's guide can be downloaded from ftp://ftp.gwu.edu/pub/graphics/VAS/.

In order to make a good sonification tool it is necessary to know more about different methods for simulating the sounds of objects. Some methods will be described in the next chapter. My own sonification toolkit is made to be used on all kinds of data and does not use any of the methods described in the next chapter. However, in a more advanced version of the toolkit it would be very interesting to implement some of these features to create realistic sounds from 3-D data of objects.

# Part III

# Simulating the Acoustics of Objects

# Chapter 5

# Acoustics of Objects

## 5.1 Approximating the Sounds of Solids

The cognitive importance of realistic sounds is well known in the entertainment industry where sampled sound effects are added to what can be seen to enhance realism. As simulations become more interactive, for instance in large architectural walkthroughs and virtual reality, synthesizing realistic object sounds directly from physical models and rendering them in real time will be increasingly important.

The sounds produced by objects may be endogenous (bark, yell, talk) which can be digitally recorded or synthesized by behavioral models (buzz from a bee determined by the way it flies), more physically based sounds emanate from vibrations of bodies caused by interactive forces between objects and their environment (collisions, friction).

Past work outside the graphics community on simulating the acoustics of solids for the purpose of generating sound has centered largely on the study of musical systems, such as strings, rigid bars, membranes, piano sound boards, and violin and guitar bodies. The techniques used include finite element and finite difference methods, lower dimensional simplifications, and modal/sinusoidal models of the eigenmodes of sound-producing systems. These approximations are flawed from a number of perspectives since most shapes are not homogenous and hence will give good results only under very specific conditions.

Sounds play a subtler role in animation and interaction (compared to alarms etc), conveying quantitative information, a sense of presence, realism, and quality. There will always be a need for hand construction of some sound effects, but for many animations it is not necessary, and for interactive simulations it is impossible since the interaction is not known in advance [40].

In many applications it would be very useful to use the same simulated motion that is already being used for generating animated video to also generate audio. O'Brien et al. in [30] managed to do this by analyzing the surface motions of objects using a deformable body simulator and isolating vibration components that correspond to audible frequencies by using known properties of solids, given be-

low. The results are said to correspond reasonably well, and the difference between the simulated results and the theoretical predictions can possibly be explained by assumptions made concerning elasticity, damping and other material features.

### 5.1.1 Known Properties of Solids

Solid objects that move in a surrounding medium induce disturbances in the medium that propagate outward. If the pressure wave is small shock waves do not form and the relationship between pressure fluctuation and density change is approximately linear. The waves are acoustic waves described by the equation:

$$\frac{\partial^2 p}{\partial t^2} = c^2 \nabla^2 p$$

where $t$ is time, $p$ is the acoustic pressure defined as the difference between the current pressure and the fluid's equilibrium pressure, and $c$ is the acoustic wave speed (speed of sound) in the fluid. The motions of the solid objects can be modeled using a non-linear finite element method. The collision forces are computed with a volume-based penalty method that allow the objects to interact with each other and with other environmental constraints. Once the pressure distribution over the surface of the objects is known the way the resulting wave propagates outward towards the listener needs to be computed.

### 5.1.2 How can a Good Physical Model be Created?

In order to acquire a large number of carefully registered measurements of a given object including shape, reflectance, sound and contact forces the process of recording and analyzing contact sounds needs to be automated. A robotic system like that should be able to acquire a model of the object shape, locate a set of points on an object to sample, move a sound measurement device to the selected points, make repeated impacts at these points, and record the resulting sounds [35].

ACME [31] is a robot made to do this to help making it easier to build more accurate and complete physical models of everyday objects.

## 5.2 Sound Rendering

Sound rendering is the process of forming the composite soundtrack from component sound objects [39]. In scientific visualization, sound can be used to characterize objects' attributes and to emphasize timing of events. The speed of sound easily causes delays perceived as echo or reverberation along with diffraction. As an object generates sound in a three-dimensional environment, it acts like a light source in image rendering. The signal propagates in all directions, is reflected and refracted by other objects and participating media, and finally caught by a receiver. The received signal is an integral of signals via multiple simultaneous paths from the emitter to the receiver. To compute this integral, each possible path can be independently

traced through the environment, its effects on the sound signal calculated, and the results composed as a convolution of the original sound [39]:

$$received(t) = \int_0^\infty w(t,\tau) \cdot emitted(t-\tau)d\tau$$

where w(t,$\tau$) is the amount of a signal received through all paths with delay $\tau$.

The sounds are treated as one-dimensional data objects associated with geometric objects of a three-dimensional world. Each sound is a time-dependent signal and extends in time instead of spatial dimensions.

According to Takala and Hahn's pioneering paper on sound rendering [39] the purpose is not to make detailed modeling of acoustics, but to show that each effect can be encoded as a transformation in the sound rendering script. The simplest acoustic effect is due to the dispersion of sound waves into the environment. The intensity decays proportionally to the square of the distance traveled (assuming no other attenuation) and the signal is delayed by a time proportional to the same distance. Lateral stereophonic hearing, due to differences in the directional sensitivity of left and right ears, can be easily simulated with two microphones facing opposite directions on a camera.

A common effect in acoustic environments is reverberation caused by reflective objects in the scene acting as secondary sound sources. In a sound script all acoustic effects of the three-dimensional world are encoded in a geometry-free form. All the complex effects can be precalculated and combined into a sequence of geometric transformations that can be concatenated into a single transformation matrix.

## 5.3 Sound Modeling

A sound model is a mathematical model that can be used to simulate the sound response of an object or for object recognition. Model parameters could include the geometry boundary conditions, mass distribution, elasticity constants for the object and its surrounding medium etc. [35].

The goal of virtualized audio is to extract sound sources from their native acoustical spaces, and insert them into a virtual acoustical space that is shared by a number of listeners. In its most general form, source separation is referred to as a blind source localization and blind deconvolution problem [13].

### 5.3.1 Approximating Contact Sounds

When an object is struck, the energy of the impact causes deformations to propagate through the body, causing its outer surfaces to vibrate and emit sound waves. The resulting sound field propagates through and also interacts with the environment. For realistic contact sounds we need good, physically based models of both the resonators and the contact interactions. The sound made by a struck object depends on many factors; object shape, location of the impact, the material of the struck object, the force of the impact etc. The two most important distinguishing characteristics of

an impact on an object are the energy transfer in the strike and the "hardness" of the contact. Theoretical models are needed to compute contact interactions for impact, rolling and sliding. Modal synthesis is a good synthesis model for solid objects since it models a vibrating object by damped harmonic oscillators, which are excited by an external stimulus. To handle contact sounds the theory needs to be extended to work well for multiple simulateneous continuous interactions at different locations with realistic timbre shifts depending on the contact point. Much work in this area has been made by van den Doel et al. [40].

Sliding involves multiple micro-collisions at the contact area. Rolling forces are, like scraping forces, produced by irregularities of the surfaces in contact, but because the surfaces have no relative speed at the contact point, the contact interactions are different and this is reflected in a difference in sound. What exactly causes the perception of rolling versus sliding is not known. Some studies suggest that the periodicity of rolling sound plays an important role in distinguishing it from sliding [40].

There are many different sound models for contact sounds, the most common being PhISM, PhSEM and PhISEM. The goal of physically informed sonic modeling (PhISM) is to couple physical simulations to efficient synthesis techniques. PhSEM (physically informed stochastic event modeling) is a PhISM algorithm based on particle models. From a psychoacoustic standpoint, the PhISEM model is appealing because it permits control over several physical parameters including the number of colliding objects, the damping properties or the resonant frequency of the objects themselves [24]. At the heart of PhISEM algorithms are particle models characterized by basic Newtonian equations governing the motion and collisions of point masses. Particle systems can be solved numberically using the same discrete differential approximations used to solve a mass/spring/damper system using the idea of particles contained in a shell [12].

## 5.4 Sonification in a Virtual Environment

The goal with sonification in a virtual environment (VE) is to process sounds so that they appear to come from particular locations in 3D-space.

The primary consideration in VE is the effective parameterization of sound models so that the parameters being generated from the motion can be mapped to them. The sound also has to be synchronized and rendered in a correct way. To render the right sound it needs to be traced within the environment. We need to take into account the listener effects such as interaural delay, shoulder reflection, head shadowing, and pinna response. These effects can be expressed as convolution filters known as Head Related Transfer Functions (HRTF) [16] described in the section below. A head-tracking device could update the directional filters in real time to compensate for a listener's head motion while virtual sources remain stable within the simulated events. To display sound being echoed requires enormous computational resources for real-time implementation [42] but the sound, and graphics, will

get better as the computer systems used becomes more advanced.

### 5.4.1 Head-related Transfer Functions (HRTFs)

According to Wenzel [42] HRTFs can be used to reproduce fully dimensional sound locations. HRTFs are empirically measured digital filters used in binaural spatialized audio systems, which simulate the direction-dependant acoustic filtering properties of the human external ear. HRTF data sets are often functions of three variables (azimuth angle, elevation angle, frequency).

Tracing sound energy within an acoustic environment can be quite complex compared to tracing light energy within a visual environment to render images [16] i.e. sound is affected by walls even outside of the virtual environment. By putting all global effects into one ambient term that is calculated by associating each partitioned space in the virtual environment with certain acoustical properties this part can be processed separately. The final sound heard is a sum of the ambient term as well as a term that corresponds to the direct path of the sound from the source to the listener.

### 5.4.2 Timbre Trees

A timbre tree is a functional representation of sounds used to model sounds that are parameterizable. These parameters can then be mapped to the parameters generated by the object motions in the environment. To use timbre trees to represent general parameterizable sounds allows any sounds to be represented and parameterized [16].

The advantage in using a tree structure is the modularity and simplicity of composing an endless variety of techniques. Nodes of the tree operate on other timbre trees, representations of sounds, or parameters. Each node of the timbre tree may represent one of the following: A numerical constant, a named parameter, a digitally sampled sound, a mathematical function with zero or more arguments, a vector of numerical constants or a reference to other timbre trees. As the objects move in the environment, the associated timbre trees move with them. The timbre tree is instantiated every time the object strikes something.

The primary requirement of any interactive use of timbre trees is that they be evaluated in real-time, which places a large computational burden on the machine. It is not unreasonable to expect that many sounds may be active in a VE at the same time and in order to achieve real-time performance with the capability of handling multiple concurrent timbre trees, parallelism can be employed.

# Part IV

# Data Sets and the VTK Sonification Toolkit

# Chapter 6

# Types of Data Sets

There are many different types of data sets, some of which will be described in this section as an introduction to the Visualization Toolkit (VTK). The data sets will be followed by more information on VTK and end with a description of my Sonification Toolkit (that uses the same data file format as VTK) in the next chapter.

It is important to make a distinction between listening to data and using data to control sound. For 1-D data sets it is possible to map the data directly to a temporal domain, like letting the data value change the pitch. However, for 2-D or 3-D data sets there are no natural mappings and we have to do the mapping from a non-temporal to a temporal domain.

There are different definitions of data dimension in AD [27, 37]. I will use McCabe's definition from [27] that mappings range from 0th to $n$th order where $n$ is the number of parameters in the audio space that are being controlled by the data.

## 6.1 Multivariate Data Sets

With a multivariate data set I will refer to a set with a large number of concurrent variables. The ability to display this data type is very important in many applications but graphical techniques of displaying multivariate data are limited in their ability to represent a large number of variables before visual clutter destroys the usefulness of the display. The addition of sonification to graphical displays of multivariate data sets may allow more aspects of the data to be presented simultaneously without creating confusion.

With multivariate data it is important to know both the relationships between various variables and which variables are predominant in carrying the information about the data [6].

**Example:** In a study performed by Fitch and Kramer [14] 13 students were to diagnose a "digital patient" and suggest treatment using visual display, AD and simultaneous visual and audio. They heard 8 different things being simulated simultaneously (blood pressure, pulse, etc.). Most students were able to treat

the patient with auditory display only but the best result was reached when vision and audio were combined.

## 6.2 Large Data Sets

Large data sets occur often in many sciences, like medicine or seismology. Sometimes the data can be successfully audified with a minimum of processing, especially when it comes to data sets displaying physics similar to that of sound. As an example seismic vibrations transmitted through the earth are somewhat similar to sounds transmitted through air [17].

**Example:** During a study by Quinn et al. [34] a collection of data from an ice core on Greenland was sonified by a simple data to pitch and duration mapping. The data provides a well-dated climate history for the past 110,000 years and consists of the 8 major ions found in the ice. This "Climate Symphony" makes it possible to hear the history of drought and flood, storms and calm, and biological growth and decay as the great continental ice sheets have advanced and retreated in the past.

## 6.3 Volumetric Data

Volumetric data include a value for a point in 3-D space, usually along with values at that specific point.

**Example:** CRUMBS is a virtual environment tracking tool for biological imaging, designed for the CAVE at the Beckman Institute for Advanced Science and Technology at the University of Illinois [1]. It allows the user to track fiber structures in volumetric data sets. The use of audio was integrated to augment the visual with melodies tied to the placement of markers.

# Chapter 7

# The VTK Sonification Toolkit (VTKST)

## 7.1 The Visualization Toolkit (VTK)

The Visualization Toolkit, VTK, is an open-source, object-oriented software system for computer graphics, visualization, and image processing [38]. Applications can be written in C++, Tcl, Python, and Java. VTK supports a wide variety of visualization algorithms, advanced modeling techniques and imaging algorithms integrated to allow the mix of 2-D imaging with 3-D graphics algorithms and data. VTK can be installed and used on Unix-based platforms, PCs and Mac (OS X Jaguar or later). VTK and documents concerning VTK can be downloaded from http://public.kitware.com/VTK/index.php. It is *not necessary to have VTK installed* to run the VTK Sonification Toolkit since it is based on the VTK file format only.

I chose to develop my sonification toolkit based on the VTK file format for the following reasons:

1. VTK is one of the most videly used tools for visualization and hence many people are familiar with the file format.

2. The file format is easy to understand and use. However it can still be used with very advanced data.

3. The file format includes all information necessary to create a useful sonification, without having to download and install VTK.

4. VTK is open source, free to download and available for most platforms (Unix, Windows, Mac) and I figured that the users of my toolkit might be interested in downloading the whole package and visualize the data being sonified.

5. I found that VTK was quite easy to learn for someone without much knowledge of the principles and theories of visualization, like myself.

6. VTK has been used in courses on KTH and it might be interesting to use VTK again together with the sonification toolkit to see whether the sonification may enhance the appreciation of the visualization.

7. Since VTK is open-source it would be possible to integrate a future version of the VTK Sonification Toolkit with VTK and coordinate a visualization with a sonification based on the same data.

### 7.1.1 The VTK File Format

A VTK file starts with a header telling what version of VTK is used and is followed by a short title and whether the data is ascii or binary. The rest of the file consists of a data set and data attributes. Please refer to appendix A for a concrete example.

#### Dataset Information

The dataset can be of six different types; structured points, structured grid, unstructured grid, poly data, rectilinear grid or a field. Field data is the most advanced since it is a general format without topological and geometric structure, and without a particular dimensionality. Field data is typically associated with the points or cells of a dataset but if it is specified as the dataset type a general VTK data object is defined.

#### Data Attributes

Each type of attribute data has a name associated with it used to identify a particular data. VTK supports the following dataset attributes: scalars, vectors, normals, texture coordinates, $3 \times 3$ tensors, and field data. Dataset attributes are supported for both points and cells.

## 7.2 The VTK Sonification Toolkit

My intention when creating a sonification model for VTK was to use the same file format for the data sets without demanding that the user had VTK installed.

Once started, the program loads the file and parses the different attributes. The parser separates the "data set" data from the "data attribute" data and puts the fields in two tables. All fields are added no matter how important they are and the user decides what to sonify and what to discard. I could have made the choice to discard data that were most probable to be uninteresting but decided to leave the choice to the user.

### 7.2.1 The VTKST classes

When creating the basic structure for the classes of VTKST I felt the need to separate the sound classes from the parsing classes and treat the mapping from data to sound in an individual class as well. Since the VTK file format is quite complicated I needed to separate the file parser into two individual classes. For the sound classes it felt natural to separate the sound properties of the computer from the MIDI events being played. To make sure everything happened in the right

order I created a main class that handles all communications with the other classes. Figure 5.1 consists of a simple class diagram.

**VTKSonificationTool** is the main class of the program. It handles the GUI and coordinates all other classes and tells them what to do. For example, once the program is started it tells the MidiHandler to check what sound capabilities the computer has and once a file has been opened it sends the message to the DataFileInterpreter that parses the file and gives the results back to the VTKSonificationTool class which displays the information in the tables (for more information on the GUI, see section 5.2.3)

**DataFileInterpreter** handles the parsing of the VTK data file and saves the information given in the file. All information is saved in hashtables for easy retrieval of the data and the related information (names, dimension, etc). When it parses a data set it opens a *DataSetFormatParser* and when it parses a data attribute it opens the *DataAttributeParser*. I decided to make two different classes for parsing data sets and data attributes since the two are the most important parts of the VTK data file and keeping them separate clarifies what the program does and simplifies the handling of the data.

**DataToSoundMapper** maps the data to sound depending on the choices the user has made. It only maps the fields that are "activated", i.e. have a mapping to either duration or pitch. It checks the minimum and maximum values in the data and maps them to the minimum and maximum values for pitch or duration set by the user and then maps the rest of the data to the corresponding values.

**MidiHandler** handles all MIDI events from the remapped data values and plays back the sound. When the program is started, the MidiHandler uses the *SynthesizerPropertyFinder* to find out if the current system has GM support, what instruments and MIDI channels are available, how many notes that can be played simultanously, etc. I chose to use MIDI because it is easy to work with and manipulate (in Java) and most computers have MIDI support (platform independent).

### 7.2.2 Sound properties

When the program is loaded it finds out dynamically what synthesizer the computer is using and what instruments are available and saves them in a list. The synthesizer information is necessary to make sure that the program can be used and that the sound can be played. Most computers nowadays have good enough sound capabilities to handle General MIDI (GM) but if the available synthesizer is not sufficient (and GM is not suported) the program will display a message and it will not start.

The user chooses what to sonify, and what to neglect, by using the drop-down menus of instruments and mappings.

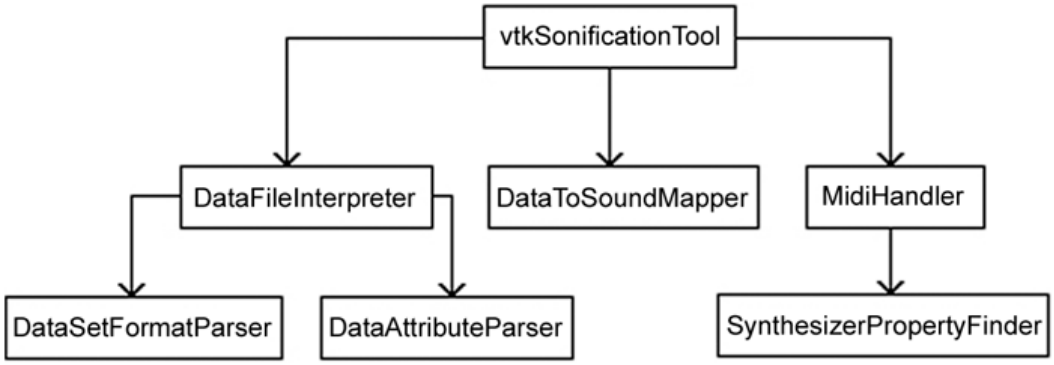


**Figure 7.1.** Class Diagram of the VTK Sonification Tool. An arrow means that the first class, where the arrow starts, creates an instance of the second class (where the arrow ends).

### 7.2.3 The Graphical User Interface

When I started thinking about the GUI for VTKST I wanted it to be simple for the user to start playing with. This was achieved by making sure the different controlling possibilities were as simple as possible to understand, and that everything displayed in the GUI was necessary. I wanted the information extracted from the file separated from the controlling mechanisms (like playing the sonification and defining minima and maxima) since it is easier to get to know a program that looks, at least partially, the same independent of which file has been opened. On the other hand, any loss in the information displayed from the data file would be unaccaptable and it is, because of the advanced VTK file format, difficult to make a really simple sonification tool. I tried to think of all those things when designing the GUI but it should be said that I do not have any formal education on designing GUIs.

Once a file has been opened two tables will be showed; one with the data set info and one with data attribute info. In the third and fourth table column the current data are mapped to an instrument and to either duration or pitch. On the right side of the program you can define minimum and maximum for pitch and duration along with default values. The different parts of the program are described in detail below. By using the menu you can save your sonification as a MIDI file or save your settings for a future session. An example of what a session may look like can be found in figure 5.2.

#### Dimension

The file parser checks the dimension of the data, and puts it in the second column of each table. The dimension of the data is necessary once the data are sonified since it would be wrong to neglect some of the data. The program can play at least 16 simultaneous sounds if it is used on a decent computer and more if it is used on a system with more advanced sound capabilities. So, if the data are multidimensional (e.g. $100 \times 3$) the program will play three sounds simultaneously. It is important

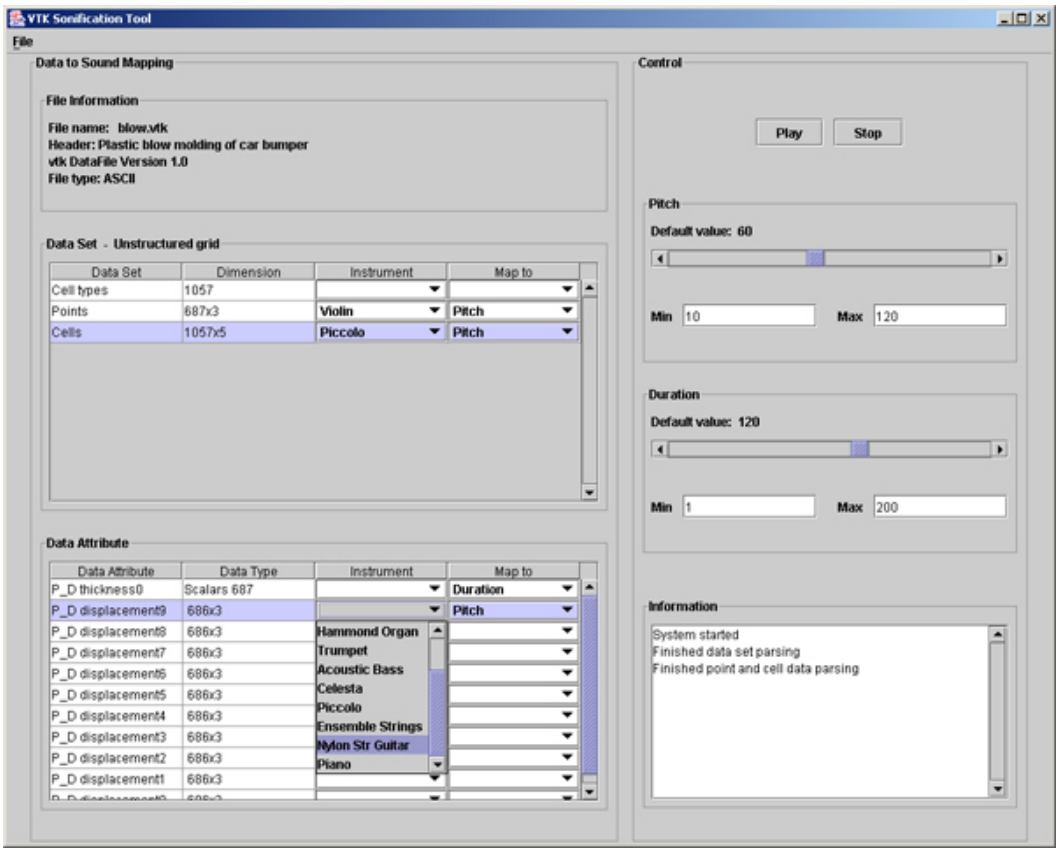


**Figure 7.2.** The VTKST GUI

to note that if a multi-dimensional data set (i.e. one row in the table) is mapped to pitch, the instrument will be the same for all sounds from the same data set. However, if a data set is mapped to duration only the first column of data will be used and the other dimensions will be neglected (read more about the consequences of mapping to duration below).

The conclusion is that if the data are multi-dimensional it is recommended that the mapping is set to pitch or else important data features from the other dimensions may be lost.

### Timbre (Instruments)

As mentioned earlier the computer's synthesizer creates a list of available instruments. On most systems there are at least 128 different instruments available. Section 3.2 tells us that we want the instruments to be as different from each other as possible and since the General MIDI (GM) specification groups the instruments into groups of eight I pick the first instrument in each group which is the main instrument of that group. If the system is not GM compatible all instruments available will be listed.

### Mapping

Section 3.2 mentioned some design guidelines which I have used to decide what mapping possibilities should be available. I found that to change the duration and the pitch are two good possibilities for a basic sonification program. With these two mapping techniques it is easy to hear the variations in the data. Volume would have been a third possibility but was quickly discarded since it would only make it more difficult to hear other features of the sonification when the volume is low and annoy when the volume is high.

If the user chooses to map the data to duration it will affect all other data sets as well since it only sets the length of the note. This is the reason that only one field can be mapped to duration and, as mentioned earlier, if the data is multidimensional it will discard all dimensions but the first. If duration is set there is no need to choose an instrument since there is no actual sound connected to the mapping. The user can choose which values are to be the minimum and maximum length of the notes. The lowest value in the data set is set to be at the minimum duration and the highest value is set to maximum duration. The data in between is mapped in a linear way to the duration values in between. The reason I chose to constrain the number of fields mapped to duration, for multi-dimensional data, is that there is really no simple way of doing it while making sure that it makes sense (consider, as an example, one row of the data being mapped to duration and the other rows to pitch—it would be virtually impossible to understand the relations from only listening to the sonification). It would be even more confusing to have many fields being mapped in this way playing simultaneously. On the other hand it is not possible to only map to duration since it is dependent on having something to play (the data values only give the duration of the note, not the actual note).

If the data are mapped to pitch an instrument should be chosen for each data set. If the data are multidimensional the program maps each dimension by itself but using the same instrument. This means that all data are fit to be between the minimum and maximum set by the user. The lowest value in the data set is set to be at the minimum pitch and the highest value is set to maximum pitch. The data in between are mapped in a linear way to the pitch values in between.

**Example 1:** The data to be sonified consist of two vectors $\vec{v}_1 = (5, 2, 10, 7, 3, 1)$ and $\vec{v}_2 = (6, 1, 2, 3, 8, 5)$. We map $v_1$ to pitch and $v_2$ to duration and set the minima and maxima to: $P_{min} = 20$, $P_{max} = 110$ and $D_{min} = 10$, $D_{max} = 20$. The result is presented in figure 5.3.

**Example 2:** If the data are multidimensional, e.g.

$$\mathbf{A_1} = \begin{pmatrix} 1 & 2 & 3 \\ 10 & 8 & 2 \\ 10 & 7 & 7 \\ 3 & 3 & 2 \\ 4 & 6 & 1 \end{pmatrix} \qquad \mathbf{A_2} = \begin{pmatrix} 2 & 3 & 8 \\ 3 & 1 & 7 \\ 1 & 3 & 2 \\ 10 & 9 & 3 \\ 4 & 1 & 6 \end{pmatrix}$$

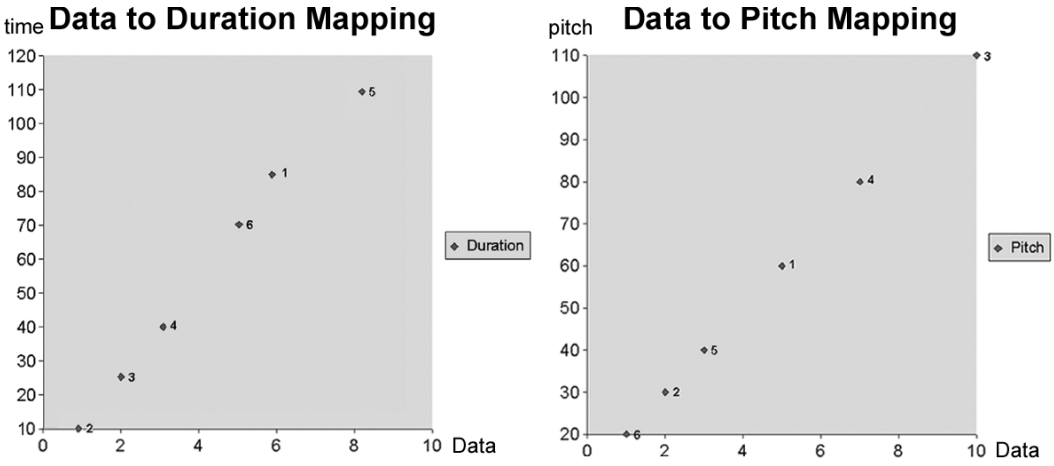


**Figure 7.3.** The numbers by the dots represent their place in the vector, e.g. the dot on $(5, 60)$ in the right diagram corresponds to the first number in $v_1$ and is mapped to 60.

and $A_1$ is mapped to pitch and $A_2$ to duration there will be three simultanous sounds (one for each column of $A_1$), all with the same duration (the first column of $A_2$, the other columns are discarded). No defaults are used in this example since all we need to know is provided. The result is presented in figure 5.4 to the left. To the right in figure 5.4 there is an example of a simple mapping to duration and pitch based on $v_1$ (pitch) and $v_2$ (duration) in example 1 above.

### Control the Sound

Once the user has decided what to sonify, the number of data fields to be sonified simultaneously, and what instruments should be used it is time to play the sonification. This is done by pressing the Play button on the top right.

Once the button is pressed the program starts mapping the data in the way mentioned above and puts each group of mapped data in a midi channel. I do not use the drum channel and hence there are 15 channels available. If there are more than 15 sounds that should be played at the same time the program tries to put more than one sound on each channel. This is not recommended for several reasons; firstly it is difficult to follow and separate more than 5–7 simultaneous sounds (as stated in chapter 2), secondly many computer systems cannot handle more than one sound on each midi channel and the result of more than one sound varies between total confusion (when trying to do too much at the same time) and only one of the sounds being played.

The user can stop the sound by pressing the Stop button. Once the sonification has been stopped changes can be made or the sonification can be saved using the menu (see details below).

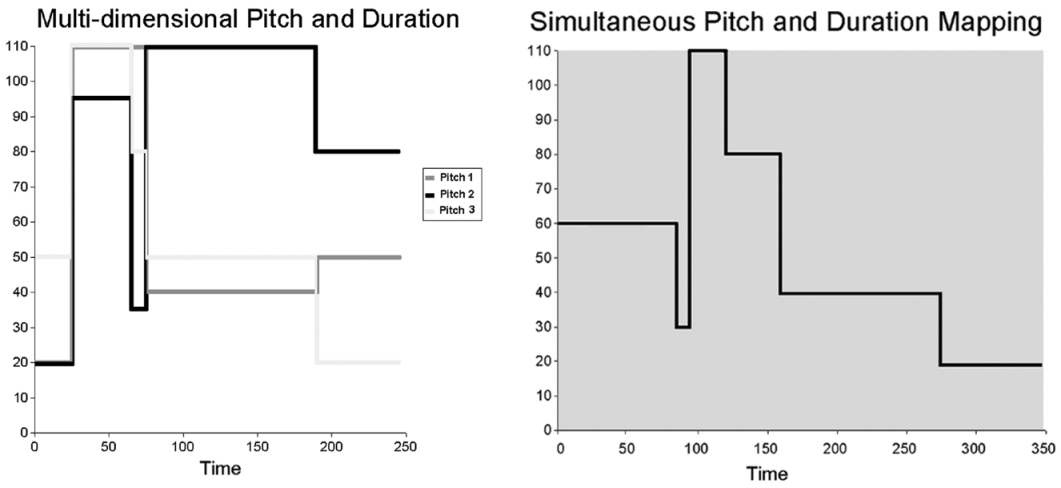


**Figure 7.4.** In the figure to the left only the first column of $A_2$ is used since it is mapped to duration. All columns in $A_1$ are mapped to pitch (but the duration of the notes is regulated by $A_2$. The x-axis represents time and the y-axis represents the pitch. This relationship is simplified in the figure to the right where I have used the values from example 1; $v_1$ (mapped to pitch) and $v_2$ (mapped to duration).

#### Information box

The information box on the lower right of the GUI will display messages of what the program is doing at the moment, if there are any errors, and possible solutions to the errors.

#### The Menu

On the top left there is a Menu that is used to open a VTK data file, save the sonification as a midi file, open settings from a previous session, save settings from an active session, get help and exit the program. When a VTK data file is opened the program starts parsing the file and creates the tables. When a sonification is saved as midi all data is saved as a midi file, the Play and Stop buttons has nothing to do with what is being saved.

It is possible to save the settings of the active session if the user wishes to remember the current settings. The data is saved as a properies file and can be reread by the program on a later occasion by using the "Open settings" option on the Menu.

### 7.2.4 Download and install VTKST

#### Download VTKST

The VTKST as a jar-file is available for download at http://www.nada.kth.se/~f97-saa/VTKST/. I have also put up some VTK files to test the program with and also some

MIDI files made with VTKST.

### Install and run VTKST

VTKST is written in Java and you will need some version of Java 1.4 installed on your system. The program is started by double-clicking on the jar file or by writing `java -jar VTKST.jar` in your terminal. The program should start by itself, a graphical interface will show and a file can be loaded and sonified.

Please note that, unlike VTK, VTKST is not able to handle binary files. In order to get VTKST to work properly it is very important the files follow the VTK format in detail.

## 7.3 Evaluation

I have made tests with all kinds of data sets supported by VTK and found that VTKST can be very useful for sonification purposes. However, as mentioned earlier it is difficult to attend to many different sounds simultanously and I have preferred to work with a maximum of 7 simultaneous sounds. The ability to work with many different instruments makes it possible to separate many different data sets played simultanously but sometimes it can be annoying with extra instruments—if I listen to two sounds mapped to different instruments it is sometimes difficult to notice whether two notes are the same or if they are very close to each other since the timbre confuses things, if I instead use the same instrument for both it is easier to notice differences in pitch.

I was able to detect some interesting features in some of the data I tested the program with, especially if I discarded some of the data fields when the file contained many data sets and if the data provided was not too complicated (some of my test files included as many as 15 different data sets and since I did not make all the files on my own I did not have much background information on the data, but used it more to test the program). E.g. I was able to hear how the melody of one data set was followed by a similar melody (in a different key) in a different data set (a bit like canon-singing) and I also noted that selecting a good tempo was a lot more difficult, and important, than it may seem. Since it is easy to change the minima and maxima for pitch and duration the differences between data values can be easily highlighted and that is usually the easiest way to find correspondances and relations between the data sets.

All sonification programs that I have read about and tried have taken some training getting used to and it is especially difficult to learn how to listen to a sonification. This may sound strange in some way since we are used to hearing meaningful sounds in everyday life, but the difference between everyday listening and listening to a sonification is huge since it takes a lot of concentration listening to, and analyzing, a sonification while the things we hear around us all the time are usually familiar sounds that we have heard before (and do not require the same amount of concentration). The sonifications made with VTKST share the problem

with the other programs (described in Part II, Software and Formats) of needing advance training, before making useful and meaningful sonifications. However, I believe that this problem can only be solved by more research in sonification and human hearing (and perception of sound), rather than making more complicated programs.

## 7.4 Suggested Changes

As I have stated in many places in this thesis a sonification is usually at its best when it is combined with another displaying technique, such as visualization. Since VTK is a visualization program it would of course be very interesting to be able to synch a visualization with a sonification for the same data. This is not possible at the moment and would be the first thing to add in a new version of the program.

The ways the data are translated to sound could be more advanced since we know more about the data than what is actually used. At the moment everything is made in the simplest possible way, and each vector is mapped directly to pitch or duration. Instead some of the methods described in previous chapters could be used to make the sonification more interesting and nicer to listen to. It would be useful to be able to create mapping functions that are especially made for the data being used. A linear mapping technique like the one being used at the moment is rarely optimal, take for example a data set with one value being 1, one 2000 and 4000 values being between 500 and 501. To make a valuable sonification under those circumstances one would want to disregard the minimum and the maximum values and use the whole pitch/duration range between 500 and 501 instead. As stated in the beginning of this chapter an integration between VTKST and VTK would be very useful in order to coordinate a visualization with a sonification based on the same data. Once VTKST is integrated with VTK it would be fairly simple to add a feature where the user makes a mapping diagram from data to sound, since this is easily done in VTK.

More instruments could be added and the way the program handles more than 16 simultaneous sounds could be made better and alternatives to MIDI should be investigated.

I will continue to work on improving the program, but this first version will have to do for now.

# References


[1] *Auditory Bread Crumbs for Navigating Volumetric Data*. Video, 1996. Produced by Rachael Brady.

[2] Robin Bargar. Pattern and reference in auditory display. In Kramer [20], pages 151–166.

[3] Stephen Barrass. Personify: a toolkit for perceptually meaningful sonification. In *Proceedings of Australian Computer Music Conference ACMA'95*. Australian Computer Music Association, Inc., 1995.

[4] Stephen Barrass. *Auditory Information Design*. PhD thesis, The Australian National University, July 1997.

[5] Merra M. Blattner, Albert L. Papp III, and Ephraim P. Glinert. Sonic enhancement of two-dimensional graphic displays. In Kramer [20], pages 447–470.

[6] Sarah Bly. Multivariate data mappings. In Kramer [20], pages 405–416.

[7] Terri L. Bonebright, Mike A. Nees, Tayla T. Connerley, and Glenn R. McCain. Testing the effectiveness of sonified graphs for education: A programmatic research project. In *Proceedings of the 2001 International conference on auditory display*, pages 62–66. ICAD, 2001.

[8] Richard Boulanger and Barry Vercoe. *CSounds*. http://www.csounds.com/.

[9] Albert Bregman. Foreword. In Kramer [20], pages xv–xxi.

[10] Stephane A. Brewster, Peter C. Wright, and Alistair D. N. Edwards. A detailed investigation into the effectiveness of earcons. In Kramer [20], pages 471–498.

[11] Jonathan Cohen. Monitoring background activities. In Kramer [20], pages 499–532.

[12] Perry R. Cook. *Real Sound Synthesis for Interactive Applications*. A K Peters, Ltd., 2002.

[13] Peter A. Dinda. Virtualized audio as a distributed interactive application. In *AccessGrid Technical Retreat 2001 Proceedings*. Argonne National Laboratory, 2001.

[14] W. Tecumseh Fitch and Gregory Kramer. Sonifying the body electric: Superiority of an auditory over a visual display in a complex, multivariate system. In Kramer [20], pages 307–326.

[15] Hesham Fouad, James A. Ballas, and Derek Brock. An extensible toolkit for creating virtual sonic environments. In Perry R. Cook, editor, *Proceedings of the 2000 International Conference on Auditory Display*, pages 32–37. ICAD, 2000.

[16] James K. Hahn, Hesham Fouad, Larry Gritz, and Jong Won Lee. Integrating sounds and motions in virtual environments. In *Sound for Animation and Virtual Reality*. Siggraph, 1995. Course Notes #10.

[17] Chris Hayward. Listening to the Earth sing. In Kramer [20], pages 369–404.

[18] Hans G. Kaper and Sever Tipei. Manifold compositions, music visualization, and scientific sonification in an immersive virtual-reality environment. In *Proceedings Int'l Computer Music Conference '98*, 1998.

[19] Hans G. Kaper, Sever Tipei, and Elizabeth Wiebel. *High-Performance Computing, Music Composition, and the Sonification of Scientific Data*. Technical report, Argonne National Laboratory, 1997.

[20] Gregory Kramer, editor. *Auditory Display, Sonification, Audification and Auditory interfaces*, volume XVIII. Santa Fe Institute, Addison-Wesley, 1994.

[21] Gregory Kramer. An introduction to auditory display. In Kramer [20], pages 1–78.

[22] Gregory Kramer. Some organizing principles for representing data with sound. In Kramer [20], pages 185–222.

[23] Gregory Kramer, Bruce Walker, Terri Bonebright, Perry Cook, John Flowers, Nadine Miner, and John Neuhoff. *Sonification Report: Status of the Field and Research Agenda*. Technical report, 1999. Prepared for the National Science Foundation by members of ICAD.

[24] Stephen Lakatos, Perry C. Cook, and Gary P. Scavone. Selective attention to the parameters of a physically informed sonic model. In *Acoustic Research Letters Online*. Acoustical Society of America, March 2000.

[25] Suresh K. Lodha, John Beahan, Travis Heppe, Abigail Joseph, and Brett Zane-Ulman. Muse: A musical data sonification toolkit. In *Proceedings of the 1997 International Conference on Auditory Display*. ICAD, 1997.

[26] Gottfried Mayer-Kress, Robin Bargar, and Insook Choi. Musical structures in data from chaotic attractors. In Kramer [20], pages 341–368.

[27] Kevin McCabe and Akil Rangwalla. Auditory display of computational fluid dynamics data. In Kramer [20], pages 327–340.

[28] *What is MIDI?* http://www.borg.com/~jglatt/tutr/whatmidi.htm.

[29] *Sonification at NCSA: Vanilla Sound Server.* http://www.ncsa.uiuc.edu/.

[30] James F. O'Brien, Perry R. Cook, and Georg Essl. Synthesizing sounds from physically based motion. In *Proceedings SIGGRAPH 2001, Computer Graphics*, 2001.

[31] Dinesh K. Pai, Jochen Lan, John Lloyd, and Robert J. Woodham. Acme, a telerobotic active measurement facility. In *Proceedings of the Sixth International Symposium on Experimental Robotics*, pages 391–400, 1999.

[32] Alex Pang, Suresh Lodha, and Craig Wittenbrink. Visualizing uncertainty in scientific data displays. In *Grantees' workshop '99.* NSF HCI Program, 1999.

[33] *Pure data dot org FAQ.* http://www.pure-data.org/.

[34] Marty Quinn. Research set to music: The climate symphony and other sonifications of ice core, radar, DNA, seismic and solar wind data. In *Proceedings of the 2001 International conference on auditory display*, pages 56–61. ICAD, 2001.

[35] Joshua L. Richmond and Dinesh K. Pai. Active measurement of contact sounds. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, 2000.

[36] Davide Rocchesso. Acoustic cues for 3-D shape information. In *Proceedings of the 2001 International conference on auditory display*, pages 175–180. ICAD, 2001.

[37] Carla Scaletti. Sound synthesis algorithms for auditory data representations. In Kramer [20], pages 223–252.

[38] William J. Schroeder, Lisa S. Avila, Kenneth M. Martin, William A. Hoffman, and C. Charles Law. *The VTK User's Guide.* Kitware, Inc, 1998.

[39] Tapio Takala and James Hahn. Sound rendering. In *SIGGRAPH 1992, Computer Graphics Proceedings*, pages 211–220, 1992.

[40] Kees van den Doel, Paul G. Kry, and Dinesh K. Pai. Foleyautomatic: Physically-based sound effects for interactive simulation and animation. In Eugene Fium, editor, *SIGGRAPH 2001, Computer Graphics Proceedings*, pages 537–544, 2001.

[41] Kees van den Doel and Dinesh K. Pai. JASS: A Java audio synthesis system for programmers. In *Proceedings of the 2001 International conference on auditory display*, pages 150–154. ICAD, 2001.

[42] Elizabeth M. Wenzel. Spatial sound and sonification. In Kramer [20], pages 127–150.

[43] Sheila M. Williams. Perceptual principles in sound grouping. In Kramer [20], pages 95–126.

[44] Catherine M. Wilson. *Listen: A Data Sonification Toolkit*. PhD thesis, University of California Santa Cruz, 1996.

[45] Fredrik Winberg. *Auditory Direct Manipulation for Blind Computer Users*. Licentiate thesis, Royal Institute of Technology, Stockholm, Sweden, 2001.

# Appendix A

# File format

This is an example of a VTK data file containing a polydata data set with 22 points and 15 lines that together spell "HELLO". The files are usually a lot more complicated with advanced data sets, data attributes, lookup tables, texture coordinates, etc.

```
# vtk DataFile Version 1.0
Stroked lines spell hello...
ASCII
DATASET POLYDATA
POINTS 22 float
0.0 0.0 0.0
0.0 2.0 0.0
0.0 1.0 0.0
1.0 1.0 0.0
1.0 0.0 0.0
1.0 2.0 0.0
2.0 0.0 0.0
3.0 0.0 0.0
2.0 2.0 0.0
3.0 2.0 0.0
2.0 1.0 0.0
3.0 1.0 0.0
4.0 0.0 0.0
5.0 0.0 0.0
4.0 2.0 0.0
6.0 0.0 0.0
7.0 0.0 0.0
6.0 2.0 0.0
8.0 0.0 0.0
9.0 0.0 0.0
8.0 2.0 0.0
9.0 2.0 0.0

LINES 15 45
2 0 1
2 4 5
2 2 3
2 6 8
2 6 7
2 10 11
2 8 9
2 12 13
2 12 14
2 15 16
2 15 17
2 18 19
2 20 21
2 18 20
2 19 21
```