WHITHER AND WHEREFORE THE AUDITORY GRAPH? ABSTRACTIONS & ÆSTHETICS IN *AUDITORY* AND *SONIFIED* GRAPHS

Paul Vickers

Interactivity Research Group
School of Informatics, Engineering, & Technology
Northumbria University
Newcastle upon Tyne
United Kingdom
paul.vickers@unn.ac.uk

ABSTRACT

A good deal of attention has been paid by the auditory display community to the sonification of graphical data and the term *auditory graph* has been used to describe this class of auditory mappings. We contend that definitions have become blurred leading to first-order sonifications of functions and data being treated as synonymous with the second- and higher-order mappings obtained when *graphs* of those functions and data are themselves sonified. This paper looks at the different types of sonifications currently known collectively as *auditory graphs* and, based on this analysis, proposes a purposeful distinction to be drawn between *auditory graphs* and *sonified graphs*. An example is taken from the domain of computer programming to further illustrate the argument.

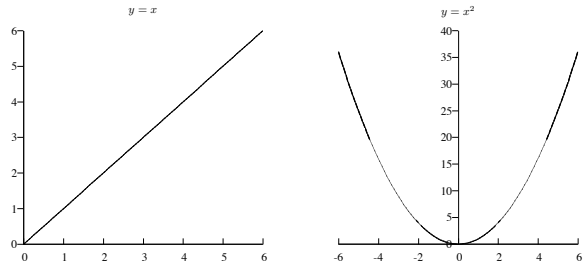
1. INTRODUCTION: WHAT GRAPH?

Since the emergence of sonification techniques for mapping data to sound, there has been much effort directed to representing graphical data using sound. For example, Lunney and Morrison [1] showed how chemical spectra could be easily communicated to blind students using simple pitch mappings. Rigas and Alty [2] demonstrated how relatively complex two-dimensional shapes could be successfully communicated to blind listeners using musical mappings alone.

More recently, researchers have begun to produce sonification toolkits that make the job of mapping data to sound relatively easy. Examples include Lodha's *Listen* and *Muse* systems [3, 4], Joseph and Lodha's *Musart* [5] and Walker and Cothran's *Sonification Sandbox* [6]. The *Sonification Sandbox* in particular is designed specifically as a tool for creating auditory graphs. In the sections that follow we explore various meanings of the term 'auditory graph' and how different interpretations of the term may be leading us to generate inappropriate or unintended sonifications.

1.1. Graphs of functions & graphs of data

At school we were taught how to graph linear and exponential functions because cartesian representations allow the overall characteristics of and differences between functions to be seen very quickly. Consider the two continuous functions $y = x$ and $y = x^2$ whose graphs (Figures 1(a) & 1(b)) clearly show their different natures. Functions with more terms and exponential powers have even more interesting graphs, and these can help the learner to understand the behaviour of functions.



(a) $y = x, 0 \leq x \leq 6$ (b) $y = x^2, -6 \leq x \leq 6$

Figure 1: Graphs of two continuous functions

In addition to plotting functions, graphs are also used (especially in the business world) to reveal features of datasets (e.g. the volume of sales in each of twelve monthly periods); data points are plotted on a graph to allow inspection and comparison of the data; indeed, the charting features of modern spreadsheet packages are designed specifically for this purpose.

1.2. Graphs as abstractions

Because graphs are so widely used it was natural that the auditory display community look for ways to map their contents to sound. This, it was claimed, would provide better analytical tools as well as making graphs accessible to those with visual impairments. A graph, though, is just an abstraction of its underlying data. The two graphs in Figure 1 are not the functions $y = x$ and $y = x^2$ themselves but external visual representations (and non-isomorphic ones at that). As they are abstractions, by their nature graphs are lossy. When we use sonification techniques to map a graph to sound we are creating a second-order abstraction (or meta abstraction) of the function and thereby potentially increasing the information loss as we move further away from the underlying data or function. When we take a tool such as the *Sonification Sandbox* to create a mapping of a graph's domain to an auditory range we are moving one step further away from the function the graph represents. Of course, the designers of the *Sonification Sandbox* state that the tool's purpose is to create direct (first-order) sonifications

of data, but the fact that the sonification is played as an accompaniment to an animated display of a two-dimensional graphical plot of the data tends to reinforce the view that the auditory graph in question is an auditory representation of the *visual* graph (see Figure 2). Thus, the very tools we use are blurring the distinction between first- and higher-order sonifications.

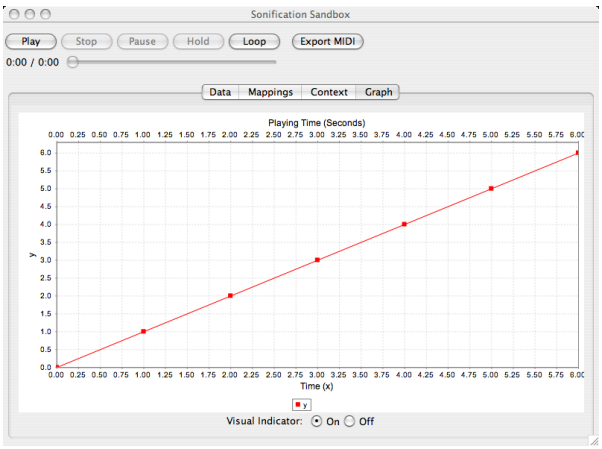


Figure 2: The continuous function $y = x$ sonified using the *Sonification Sandbox*

The auditory graphs generated by existing tools tend to produce sonifications that map the contour of the visual graph to sound. This is fine for functions with one term, such as $y = x$ and $y = x^2$, but what about those with more terms? Take the generalised quadratic function $f(x) = ax^2 + bx + c$ or a sinusoidal function such as $f(t) = t.sin(\omega t)$. A sonification of the corresponding graphs using a tool such as the *Sonification Sandbox* only plays the values of y for each value in a specified interval of x: what about the other terms and coefficients? We could, of course, direct the *Sonification Sandbox* to map the constants a, b, and c to sound (though this would not be very instructive), but what about the variable terms ax^2 and bx? By mapping y to the auditory domain we are getting an auditory graph of the entire function where it may be instructive to hear the individual terms. Again, we could get round this by creating more columns of data in the *Sonification Sandbox* to represent ax^2 and bx as terms in their own right and play all the sonfications in parallel, but this seems like a lot of work. What is needed is a tool that takes a function or a dataset and allows the properties of the data to be explored sonically in a more natural way (that is, one that is more consonant with the underlying data or function).

2. DESIGN ÆSTHETICS: THE CURSE OF MIDI

Sonification toolkits generally use MIDI as the medium for turning data into sound. A fundamental restriction of MIDI is that by quantizing the data to fit the 128 available MIDI pitches it effectively turns continuous data or functions into discrete forms. What does this do to the perception of the graph and the listener's understanding of it? Take the continuous function $y = x$ whose graph is shown in Figure 1(a) for the interval $0 \leq x \leq 6$. This is a simple *straight line* function[1], and we can use the *Sonification Soundbox* to generate an auditory graph (see Figure 2). The *Sonification Soundbox* works with tabular data, and so the function $y = x$ must be mapped from the continuous to the discrete domain to give the data pairs {{1,1},{2,2},{3,3},{4,4},{5,5},{6,6}}. These data pairs are then transformed via a mapping function inside the *Sandbox* into MIDI note-on/note-off events which are in turn executed by a software synthesiser. The output (see Figure 2) will be heard as a series of six rising pitch intervals (the exact interval between pitches is not specified but is itself a function of the number of data points in the table and the minimum and maximum pitches that have been set by the user). Using techniques like this Walker and Mauney [7], Brown and Brewster [8], and others have demonstrated that such auditory mappings can be understood by listeners and different graphs discriminated by their auditory signatures. But consider what has happened. We began with a function, $y = x$, and carried out a transformation that allows a specified interval of x to be represented in discrete tabular form (a first-order abstraction which we may call A). The discrete data points are themselves quantized to MIDI note numbers (giving the second-order abstraction A') which are in turn quantized to actual audible frequencies (A''). Finally, the MIDI note numbers themselves are a function of the data and the pitch range (A'''). This means that the auditory graph is as much as a fourth-order abstraction (a meta-meta-meta-abstraction) of the original function.

The quantization imposed by the MIDI system means that, in the case of $y = x$, the auditory representation is of a *step function* rather than a continuous function. Thus the function $y = x$ (Figure 1(a)) is rendered by *Sonification Sandbox* (Figure 2) as an auditory mapping of the greatest integer function $y = [x]$ (Figure 3). The quantisation effect is shown more clearly in Figure 4 in

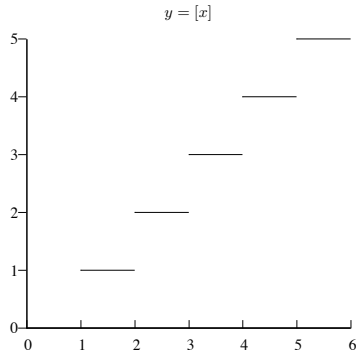


Figure 3: The Greatest Integer function: $y = [x]$

which the function $y = x^2$ has been converted to discrete tabular data and sonified using the *Sonification Sandbox*. The quantisation steps can be masked somewhat by using the *Sandbox*'s interpolation feature to produce a smooth *glissando* effect between the discrete MIDI pitches. However, the limitations of MIDI's resolution, and the requirement to trigger specific Note-on/Note-off events for each data point tone, means that the quantisation steps can still be clearly heard. Careful selection of timbre ameliorates this effect still further, but does not remove the problem altogether.

Whilst MIDI makes the building of sonification tools easier, it does not lend itself well to rich auditory displays. It is, for the most part, tied to the twelve-tone octave and does not contain the vocabulary necessary for detailed expressive control of auditory events.

[1] Note the way functions are known by the shape of their graph.

The æsthetic of MIDI lies predominantly in the keyboard-based synthesiser technology of the 1980s and so its expressive controls are modelled around those found on keyboard-based instruments[2]. Tools such as *Max/MSP*[3], *Pure Data*[4] and *SuperCollider*[5] provide much more flexible environments that open up the æsthetic possibilities of sonification tools. There is no reason now for sonifications to confine themselves to keyboard-instrument-based tonal (or even atonal) frameworks if the æsthetics of other auditory forms and languages lend themselves better to the task in hand. Whilst there are difficulties associated with straying too far from the listener's own frame of reference [9] there is a need to move beyond the strictures imposed by the MIDI protocols.

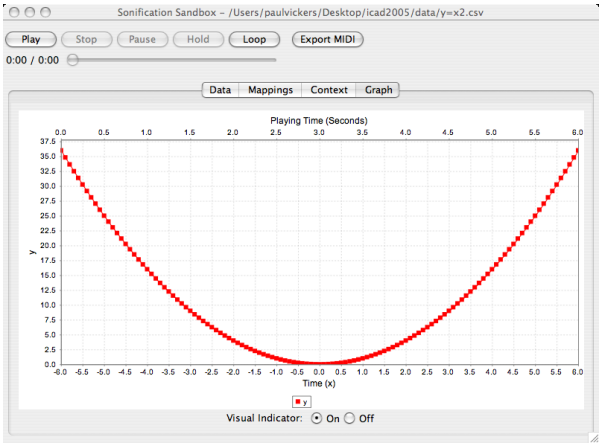


Figure 4: The discrete form of the continuous function $y = x^2$ sonified using the *Sonification Sandbox*

3. WHEREFORE THE AUDITORY GRAPH?

Perhaps the term *auditory graph* is itself misleading. The examples above have interpreted the term as meaning an auditory representation of a graph. Indeed, Brown and Brewster [8] used the term *sonified line graph* explicitly to mean a sonification of a visual graph. However, this interpretation does not always hold. The *Sonification Sandbox* creates sonifications of tabular data. As long as these data are not themselves abstractions of something else, then the resultant sonification is a first-order external auditory representation of the data in the same way that a graph or chart is a first-order visual representation of the data. In this sense the term *auditory graph* is a metaphor that attempts to show how the sonification is *like* a graph only rendered in sound rather than graphically. Here, the auditory graph is *not* a mapping of the graph's domain onto an auditory range but a mapping of the function or data domain to an auditory range. However, the blurring of definitions is compounded when the *Sandbox* plays the sonification in sync with an annotated graph of the data whose x axis is traversed by a cursor in time with the sonification (see Figure 2).

[2]We are aware of MIDI breath controllers and the like, but the resolution of the MIDI language severely restricts their usefulness.

[3]see http://www.cycling74.com/products/maxmsp.html

[4]see http://www.puredata.org

[5]see http://www.audiosynth.com

3.1. Auditory graphs and sonified graphs

It would be helpful to have different terms for these different types of sonification. Therefore, we propose that the term *auditory graph* be reserved for those first-order sonifications of data and functions that are thought of as being analogous to a visual graph. When a graph itself is mapped to sound (as was the case in Brown and Brewster's work [8]) then, to borrow from Brown and Brewster's terminology, we are dealing with a *sonified graph*—see Figure 5.

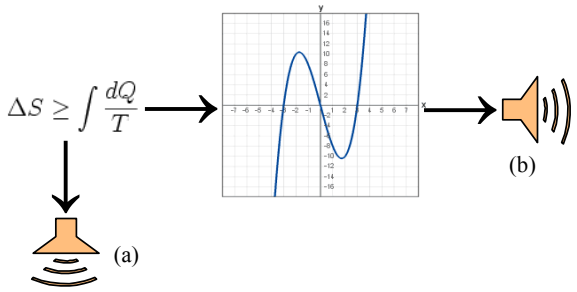


Figure 5: Two different meanings of the term *auditory graph*. (a) A *real* auditory graph: Mapping a function to sound. (b) Sonified graph: Mapping a *graph* to sound

4. EXAMPLE FROM THE PROGRAMMING DOMAIN

Graphs are common in the world of programming. Consider Figure 6 which shows the structure of an algorithm for handling a library book using Jackson's tree notation [10]. The tree diagram is one of many external visual representations based upon graph-theoretic foundations that are used in programming. Any

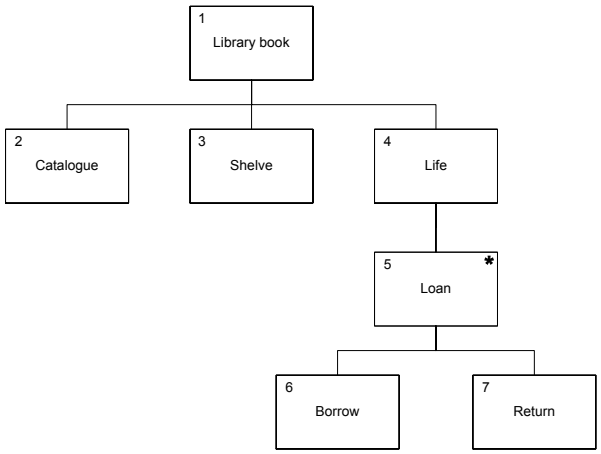


Figure 6: Tree graph for library book

Jackson tree diagram can be transformed into a state-transition diagram (finite state machine). Figure 7 shows a state transition diagram transformation of the tree diagram from Figure 6. Both notations in Figures 6 and 7 are graphs, but they emphasise different aspects of the underlying algorithm. Figure 6 emphasises the algorithm's structure whilst Figure 7 focuses on the events that

cause change in the system. If the compound nodes on the tree in Figure 7 are broken into two: 'b' for start and 'e' for end, then we can also derive a directed graph with ten edges and ten vertices:
$V := \{1b, 1e, 2, 3, 4b, 4e, 5b, 5e, 6, 7\}$
$E := \{\{1b, 2\}, \{2, 3\}, \{3, 4b\}, \{4b, 5b\}, \{5b, 6\}, \{6, 7\}, \{7, 5e\}, \{5e, 4b\}, \{4b, 4e\}, \{4e, 1e\}\}$
which can be represented diagramatically as Figure 8.

Taken together, Figures 6, 7, and 8 all provide different first-order

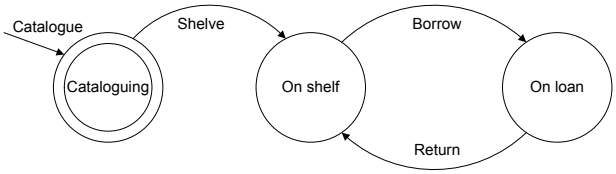


Figure 7: Finite state machine for library book

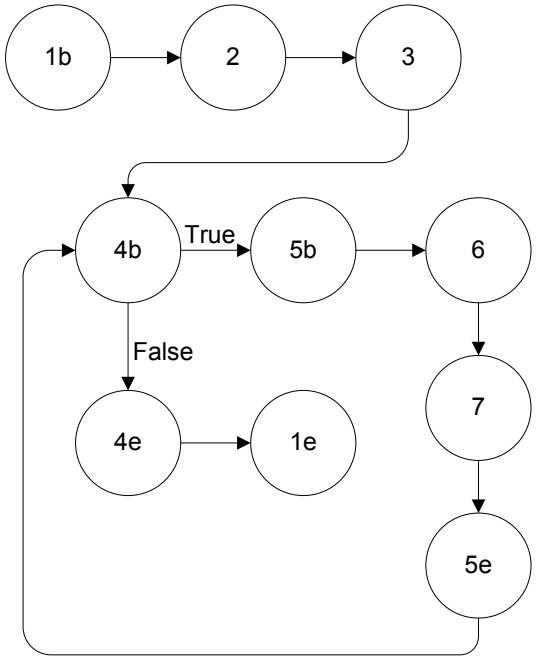


Figure 8: Directed graph for library book

representations of a shared underlying algorithm allowing insights into its structure and behaviour. Vickers and Alty [11, 12, 13] demonstrated that the first-order sonifications (or *auralisations*) of programs generated by their `CAITLIN` system could be used in the identification and location of bugs. Using the above definitions of *auditory graph* and *sonified graph*, the auralisations generated by `CAITLIN` are first-order auditory graphs, just as Figures 6, 7, and 8 are all first-order visual graphs. In an earlier work Vickers [14] commented that the auralisations were like auditory tree diagrams, but close inspection shows that this is more of an analogy than anything else. The auralisations were sonifications of the run-time execution path of the program and did, indeed, contain structural information like a tree diagram, but as they were not, themselves, sonifications of tree diagrams, the analogy broke down beyond a certain level of abstraction.

The `CAITLIN` program auralisation system, then, generated true auditory graphs which, being first-order external representations of the underlying program, sit alongside the external visual representations (trees, finite state machines, directed graphs, etc) to provide an audio-visual tool set for the exploration of programs. Each tool in the set provides a different view onto the program. However, sonified graphs of the trees, finite state machines, and directed graphs, would provide auditory views onto the *visual graphs*. The sonified graphs would thus be first-order representations of the visual graphs and second-order representations of the underlying program.

5. WHITHER THE AUDITORY GRAPH?

How, then, might the community progress from the 'simple'[6] mappings employed in the sonification of visual graphs to the richer mappings needed to allow fuller exploration of functions and data sets? Some lessons may, perhaps, be learnt from program auralisation work: auralisations can render information that is harder to spot using visual representation systems (e.g. message passing between parallel threads). Another angle of attack may come from the work of Hermann and Ritter [15] who proposed treating a dataset as a virtual physical entity from which a physical model is derived. The model is then excited to produce sound (just like physical models of musical instruments). This approach allows a form of data mining to be carried out on the dataset and reveals different features of the data sonically.

We suggest that the future of auditory graphs may lie in these kinds of approaches. Rather than simply trying to create auditory equivalents of visual graphs, treating datasets and functions holistically will enable auditory mappings that *naturally* correspond to their features and characteristics. If a function is treated as the specification for a physical model then much richer sonifications could be derived than the simple transliteration of a function's graph contour into sound allows. Like the `CAITLIN` program auralisations the auditory graph then becomes a true external representation in its own right giving a different perspective on its underlying data and complementing rather than reproducing visual graphs. Sonified graphs can be retained as alternatives to visual graphs but in the full knowledge that they are not first-order sonifications.

6. CONCLUSIONS

The auditory display community should be careful to distinguish between *auditory graphs* (first-order sonifications of data) and *sonified graphs* (sonifications of graphs). If we go down the road of focusing our attention on sonified graphs we may miss out on the full potential of auditory display. Back in 1990 Hotchkiss and Wampler [16] wrote:

> Humans have been so accustomed to looking at graphs of functions that a great richness of understanding has been sorely overlooked.

Having concentrated much effort on sonifying graphs, the auditory display community should now focus attention on the mappings that will create true auditory graphs that enable data and functions

[6] Simple in the sense that a good sonified graph should aim to be, in information terms at least, as close to an isomorphic mapping of the visual graph it portrays as possible. A true sonified graph should not aim to present more information than is present in the visual graph it represents.

to be explored in the ways Hotchkiss and Wampler talked about. Of couse, the sonified graph has its place, but by clearly marking the difference between *auditory* and *sonified* graphs we can begin to exploit the auditory channel more effectively.

7. REFERENCES

[1] D Lunney and R C Morrison, "Auditory presentation of experimental data," in *Extracting Meaning from Complex Data: Processing, Display and Interaction*, E J Farrell, Ed., vol. 1259, pp. 140–146. 1990.

[2] Dimitrios I Rigas and James L Alty, "The use of music in a graphical interface for the visually impaired," in *Interact '97*, S. Howard, J. Hammond, and G. Lindegaard, Eds., Sydney, 1997, pp. 228–235, Chapman & Hall, London.

[3] C M Wilson and Suresh K Lodha, "Listen: A data sonification toolkit," in *ICAD '96 3rd International Conference on Auditory Display*, S Frysinger, Ed., Palo Alto, USA, 1996.

[4] Suresh K Lodha, J Beahan, T Heppe, Abigail J Joseph, and B Zne-Ulman, "Muse: A musical data sonificaton toolkit," in *ICAD '97 4th International Conference on Auditory Display*, Palo Alto, USA, 1997.

[5] Abigail J Joseph and Suresh K Lodha, "Musart: Musical audio transfer function real-time toolkit," in *ICAD '02 - 2002 International Conference on Auditory Display*, Kyoto, Japan, 2002.

[6] Bruce N Walker and Joshua T Cothran, "Sonification Sandbox: A graphical toolkit for auditory graphs," in *ICAD '03 9th International Conference on Auditory Display*, Boston, MA, USA, July 2003, pp. 161–163.

[7] Bruce N Walker and Lisa M Mauney, "Individual differences, cognitive abilities, and the interpretation of auditory graphs," in *ICAD '04 The 10th Meeting of the International Conference on Auditory Display*, Steven Barrass and Paul Vickers, Eds., Sydney, Australia, July 6-9 2004.

[8] L M Brown and S A Brewster, "Drawing by ear: Interpreting sonified line graphs," in *ICAD '03 9th International Conference on Auditory Display*, Boston, MA, USA, July 2003.

[9] Paul Vickers and James L Alty, "The well-tempered compiler: The aesthetics of program auralization," in *Aesthetic Computing*, Paul Fishwick, Ed., chapter 11, p. in press. MIT Press, Boston, MA, 2005.

[10] M A Jackson, *Principles of Program Design*, London:Academic Press, 1975.

[11] Paul Vickers and James L Alty, "Using music to communicate computing information," *Interacting with Computers*, vol. 14, no. 5, pp. 435–456, 2002.

[12] Paul Vickers and James L Alty, "Musical program auralisation: A structured approach to motif design," *Interacting with Computers*, vol. 14, no. 5, pp. 457–485, 2002.

[13] Paul Vickers and James L Alty, "When bugs sing," *Interacting with Computers*, vol. 14, no. 6, pp. 793–819, 2002.

[14] Paul Vickers, *CAITLIN: Implementation of a Musical Program Auralisation System to Study the Effects on Debugging Tasks as Performed by Novice Pascal Programmers*, Ph.d. thesis, Loughborough University, Loughborough, Leicestershire, September 1999.

[15] Thomas Hermann and Helge Ritter, "Listen to your data: Model-based sonification for data analysis," in *Advances in intelligent computing and multimedia systems*, Baden-Baden, Germany, 1999, pp. 189–194, Int. Inst. for Advanced Studies in System research and cybernetics.

[16] Robert S Hotchkiss and Cheryl L Wampler, "The auditorialization of scientific information," in *Supercomputing*. 1991, pp. 453–461, ACM Press.