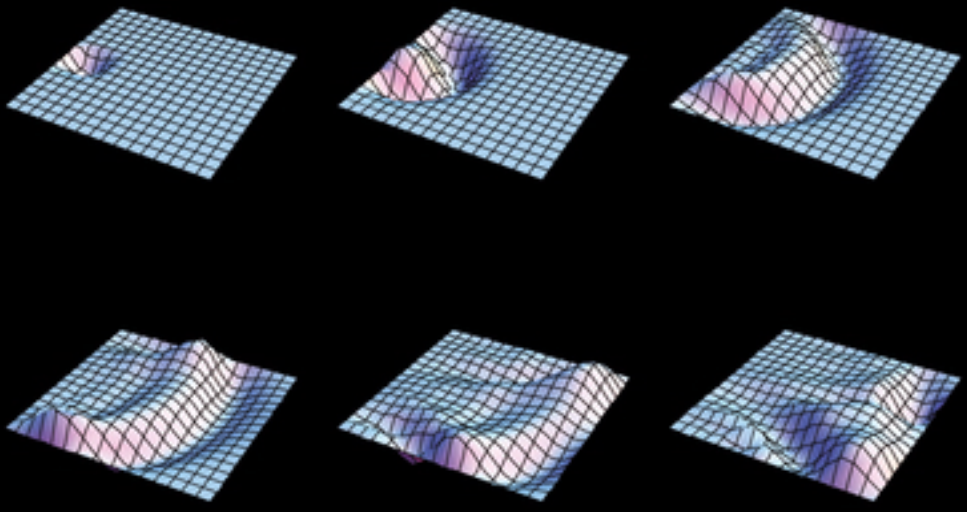
# Signals, Systems and Sound Synthesis

Martin Neukom



Peter Lang

# 1 How to Use This Book

## 1.1 Getting Started

This book is a translation from the German original, *Systeme, Signale und Klangsynthese* (2003, 2005). The text for the English edition has been considerably revised, and whole sections have been rewritten. It has some specific features of format and layout that distinguish it from most other books, largely due to the fact that it is written using the computational software *Mathematica*. While graphically not as flexible as digital publishing software, Mathematica allows the text to be published simultaneously as a book and as a CD containing the complete text in computer-readable form, many directly executable programs as well as links to additional relevant passages within the text, example programs, animations and sound examples.

The book is stored on the CD in so-called *Notebooks* (see 1.3 below). It was decided that the printed text and the screen version of the Mathematica notebooks should look as much alike as possible. Because notebooks have no distinct pages, and hence no page numbers, reference within the book is not made to pages but to chapters and sub-chapters. The illustrations always follow immediately their explanations in the text and therefore have no captions. The mathematical formulas are not numbered as usual but instead are linked or repeated where necessary.

In the German original, Chapter 1 (this chapter) and the introductions to the programming languages Csound and C were quite long. In this edition these chapters have been drastically shortened since both programming languages are well documented in English. In Chapter 4 one will find short descriptions of the programming languages one will encounter in the text as well as explanations of some special applications and procedures used in the book.

The bibliography of the original edition contained a commented list of German and English books suitable for interested non-professional readers. In this edition, the books listed have been updated to their most recent editions, and most German references have been replaced by their English versions where these exist or by comparable English language literature. Most of the methods, formulas and algorithms used here belong to the common practice of their respective disciplines, and their origins cannot be determined with certainty. For this reason, no texts are quoted in the book and in general no mention is made of authors of formulas or techniques. In addition, reference is usually made to secondary literature and not to primary literature.

The CD contains:

– The complete text of the book as Mathematica-notebooks (*name*.nb) in the folder *Text*;
– Additional chapters, explanations and examples as hidden cells in the notebooks;
– C/C++ programs in source code (*name*.cpp) in the folder *CPP*;
– Java programs in source code (*name*.java) and Java classes (*name*.class) for the mxj-externals in the folder *Java*;
– Sound examples (*name*.wav) in the folder *Sounds*;
– Csound examples in the folder *Csound*;
– Max patches in the folder *Max*;
– Programs in the Processing language in the folder *Processing*.

## 1.2 Overview

This book gives an introduction to the techniques of digital sound synthesis and sound transformation. The relevant topics are presented using illustrations, animations of complex physical and mathematical relationships, sound examples and sample programs. Basic technical and mathematical principles will be explained where they are necessary for reading the specialized literature.
This first chapter gives an overview of the book's contents, the enclosed CD and the computer programs used in the book. In Chapters 2 to 4 the physical, mathematical and programming essentials for the rest of the book are developed. The short overview of acoustics in Chapter 2 is followed in Chapter 3 by a thorough presentation of signal theory and system theory. These theories provide the tools for a precise derivation of many techniques of sound synthesis, sound transformation and control theory. The text is written to give a clear and intuitive view of the material rather than a more abstract, general presentation. The chapters that follow are written so they can be understood in their broad lines without the theoretical material developed in Chapter 3. Traditionally, many programming languages and specific programs have been used in the treatment and synthesis of sound. In Chapter 4, only those languages are discussed which are later used in the book. Mathematica was chosen for the body of the text, because an editor, a high-level programming language and routines to generate illustrations, animations and sounds are integrated into the language. C/C++ was chosen as a general programming language because Csound, Max and many other programs are written in C and can be extended using additional routines written in C. Csound was chosen as the domain-specific language (DSL) for sound synthesis because it has a long history in computer music and because it has the flexibility of a general programming language. Max was chosen for interactive programming because it is the most frequently used language in live electronics. Max is intuitive to use and ideal for demonstrations because of its graphic interface. Several techniques of sound synthesis and sound treatment are introduced in Chapters 5 to 8. Nonlinear techniques and techniques that simulate the physical procedures of natural sound production are particularly emphasized because they give interesting results with only simple programming and modest computation times. Chapter 8 gives a comprehensive introduction to sound synthesis by physical modeling. At the same time it offers a first taste of digital filter theory. Chapter 9 discusses some of the many problems that arise in connection with the synthesis of sound in resonant spaces and presents suggestions for solutions. Chapter 10, finally, discusses techniques and aids for the composition of computer music.

## 1.3 Instructions for Using Specific Programs

The rest of this chapter contains instructions for using the data and the programs on the enclosed CD. Chapter 4 contains more detailed descriptions of the programs and programming languages Mathematica, C/C++, Csound, Max and Processing.

The text of this book is stored in Mathematica notebooks, which can be read with the *Wolfram CDF Player* or read and edited with the program *Mathematica*. The program Wolfram CDF Player can be downloaded from Wolfram Research (http://www.wolfram.com). The notebooks include closed *cells* containing additional text, the programs used to generate the illustrations, animations ( → Animation) and sound examples ( → Sound example) as well as additional cells. Longer sound examples are stored on the CD as WAVE files.

The *Csound program* (Version 5 or later) together with *Csound scores* and *orchestras* generates sounds. All the sounds synthesized by the Csound program in the course of the book can be found on the enclosed CD.

All the *C/C++ programs* can be compiled on a C++ compiler, but only some of the examples can be compiled on a classical C compiler.

The *Max programs* can be run using the freely available program *Max Runtime*. They can be executed, edited and extended using the Max program itself (http://cycling74.com).

The examples written in the language *Processing* can be executed, edited and extended using the freely available program Processing (http://processing.org).

### 1.3.1 Using the Mathematica Notebooks

After downloading and installing the Wolfram CDF Player, start the program. Now a notebook can be opened using the Menu item *File*. If the notebook *Contents.nb* or *Index.nb* is opened first, the other notebooks can be opened using hyperlinks (see below). A Mathematica notebook consists of cells that contain text, illustrations, sounds or other cells. The cells are indicated by brackets on the right edge of the screen. Nested cells are shown by nested brackets. By double-clicking on an outer bracket, whole groups of cells can be closed so that only the uppermost cell is visible. By double-clicking on an inner cell bracket, a group of cells can be closed so that only the selected cell is visible. Closed cells can be opend by double-clicking on a bracket with a hook. The text appears as in the printed book. The hidden cells contain the programs used to generate illustrations, animations and sounds as well as calculations and additional texts.

To start an animation, open the the group of cells whose visible cell has the comment ( → Animation) by double-clicking on the corresponding bracket. Most animations are interactive. Their parameters can be changed by graphic elements in and sliders next to the illustrations. The sliders can be animated by clicking on the plus sign next to the slider. In this way continuous animations can be generated.

The CD version of the book contains hyperlinks which give access to other cells in the current notebook or to cells in other notebooks. These hyperlinks are colored blue. Clicking on a hyperlink retrieves the corresponding cell.

### 1.3.2 Using the Csound Programs

Csound is a command-line oriented programming language designed for synthesizing and manipulating sound. Its name comes from the fact that it is written in C. For the examples in this book *QuteCsound* was used as a development environment within which programs can be edited and executed. QuteCsound has a highlighting editor with autocompletion, interactive features and integrated help files. In the Csound folder the orchestra and score programs have been combined to form files in so-called *Csound unified file format* (*name.csd*).

### 1.3.3 Using the C/C++ Programs

C++ is an intermediate-level general-purpose programming language. It contains nearly all the features of the language C and adds object-oriented features, in particular, classes. This

book does not always distinguish explicitly between programs in C and programs in C++. All the programs can be compiled with a C++ compiler, but only some can be compiled with a C compiler. In order to execute the programs that write sound files, the header file my_WAVE.h must be included in the program text using the command: #include “my_WAVE.h”.

### 1.3.4 Using the Max Patches

The following explanations can be tested interactively by double-clicking on the Max patch *Instructions.maxpat* in the folder Max. Max patches are made up of elements such as generators, faders, inputs and outputs, etc. They can also contain subpatches. The audio settings can be changed by clicking on the word “Audio”, which opens the window “DSP Status”. The settings used for the examples in the book are stored under “Presets”. Parameter values can be changed in the corresponding number fields and graphic elements and can then be stored as new presets. Clicking on the subpatch *p name* opens the subpatch and shows it in a new window. The subpatch *p comments* tells what the respective Max patch does and how to use it. The subpatch *p presets* explains the preset settings. Max objects can be programmed in C or Java as so-called *externals*. In this book only Java externals are used. Double-clicking on the object *mxj quickie name* displays the source code of the external *name*.

### 1.3.5 Using the Processing Programs

The examples written in the language Processing are stored as source code on the enclosed CD and must be executed using the freely available program Processing . Some examples require libraries that are not automatically installed with Processing (controlP5, oscP5, netP5) (http://processing.org). Instructions for and comments on the Processing examples can be found in the headers of the respective source code files.

# 2 Fundamentals of Acoustics

*Further Reading:* Musical Acoustics *by Donald E. Hall [20] gives a comprehensive introduction to the general topic*. Music Cognition and Computerized Sound *by Perry R. Cook [41] is especially useful because of the enclosed CD with sound examples. The fundamentals of acoustic are summarized in several books on computer music, among them* Computer Music *by Charles Dodge and Thomas A. Jerse [2]*. Genealogie der Klangfarbe *by Daniel Muzzulini [77] presents a detailed discussion of the phenomenon of timbre*.

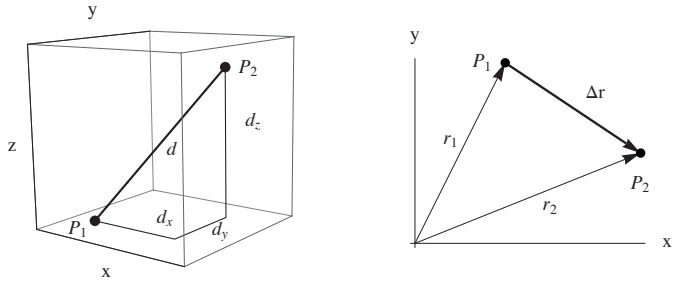
## 2.1 Basic Physical Principles and Units

### 2.1.1 Path, Velocity, Acceleration

The distance between the points $P_1$ and $P_2$ is $|x_2 - x_1| = |x_1 - x_2|$. In Cartesian coordinates (straight perpendicular axes with the same unit of measurement) in two or three dimensions, the Euclidean distance $d$ is calculated from the difference of the coordinates by

$$d = \sqrt{d_x^2 + d_y^2} \text{ or } d = \sqrt{d_x^2 + d_y^2 + d_z^2}$$

respectively (left figure below). If the points are defined by position vectors $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ from the origin, then the *shortest path* $\Delta\boldsymbol{r} = \boldsymbol{r}_2 - \boldsymbol{r}_1$. The formulas above give the corresponding distance (right figure below).




If it takes time $\Delta t$ for a point to traverse the path $\Delta\boldsymbol{r}$, then the point's mean *velocity* is $\bar{\boldsymbol{v}} = \Delta\boldsymbol{r}/\Delta t$. The direction of $\Delta\boldsymbol{r}$ and $\bar{\boldsymbol{v}}$ are the same. The instantaneous velocity $\boldsymbol{v}$ is equal to the limit of $\Delta\boldsymbol{r}/\Delta t$ when $\Delta t$ approaches 0, that is, equal to the first derivative of the position with respect to time: $\boldsymbol{v} = \lim_{\Delta t \to 0}(\Delta\boldsymbol{r}/\Delta t) = d\boldsymbol{r}/dt$. In contrast to the vector velocity $\boldsymbol{v}$, we refer to the magnitude of the velocity as speed and write $v$ or $u$. *Acceleration* is defined as the change in velocity over time: $\boldsymbol{a} = \Delta\boldsymbol{v}/\Delta t$. The instantaneous acceleration is defined as: $\boldsymbol{a} = \lim_{\Delta t \to 0}(\Delta\boldsymbol{v}/\Delta t) = d\boldsymbol{v}/dt$. Correspondingly, velocity is calculated by integrating acceleration over time, position by integrating velocity over time (A5.3).

In the following example a mass is dropped from a height of 100 m. We know the position $x = 100$ m at time $t = 0$ s, the speed $v = 0$ m/s at time $t = 0$ s and the acceleration given by gravitation $g$. If the $x$-axis is pointing upward (figure to the right), the acceleration is negative: $g \approx -10$ m/s$^2$. Since the acceleration is constant, we calculate from $a = dv/dt$ the speed $v = tg$. For the speed after 3 seconds, we have

$$v(3) = -3 \text{ s} \cdot 10 \text{ m/s}^2 = -30 \text{ m/s}.$$