

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
Department "Institut für Informatik"
Lehr- und Forschungseinheit Medieninformatik
Prof. Dr. Heinrich Hußmann

Diplomarbeit

AudioPhield:

Exploring Casual Collaborative Browsing of Large Music
Collections

Matthias W. Schicker
schicke@ifi.lmu.de

Bearbeitungszeitraum: 2. 3. 2008 bis 16. 9. 2008
Betreuer: Dipl. Inf. Otmar Hilliges
Verantw. Hochschullehrer: Prof. Dr. Andreas Butz

Acknowledgments

No diploma thesis is written alone. Well, technically, this paper *was* written alone, but it was *shaped* by more people than just me. And this is the place to thank them for it.

First of all, Otmar Hilliges, my tutor: His (and mine) interest and enthusiasm caused several redesigns from scratch – and heightened the feeling that I was working on something really cool. His input and opinions are also the cause for the “we” throughout this work.

Then I would like to thank the whole Media Informatics group for even more input and looking sincerely awed at one of AudioPhield’s demos. From this crowd, I thank especially Heiko Drewes, for helping me with fisheye lenses, and Dominikus Baur and Richard Atterer, who had to endure the sanity check of the evaluation and thus spent more of a Friday evening in university rooms than they deserve.

A big “thank you” goes also to the participants of my user study for their time, music and applause.

Of course, I want also to thank all the people that may have not been directly involved in this work but have enabled it nonetheless. Thank you for being supportive and for enduring all the panic and foul mood before deadlines.

And yes, even if every word above is true, this acknowledgment is also a humble effort to mellow my correctors, of course.

Zusammenfassung

Musik spielt eine wichtige Rolle im privaten und sozialen Leben des modernen Menschen. Dies involviert das Fachsimpeln über Musik und die gemeinsame, explorative Suche nach geeigneter Musik für einen bestimmten Anlass oder eine bestimmte Stimmung. Herkömmliche Systeme zur Verwaltung und Durchforstung von Kollektionen digitaler Musik sind zur Unterstützung dieser Tätigkeiten ungeeignet.

Deshalb präsentieren wir in dieser Arbeit das für Tabletop Displays bestimmte Softwaresystem „AudioPhield“. AudioPhield stellt die einzelnen Songs aus den Sammlungen verschiedener Benutzer gleichzeitig auf einer Starfield-artigen Benutzeroberfläche dar. Dabei sollen Stücke, die als ähnlich wahrgenommen werden, auch nah beieinander platziert werden.

Der Fokus der Arbeit liegt auf der Visualisierung der Musiksammlungen und der Entwicklung geeigneter Interaktionsformen, die es mehreren Benutzern erlauben, das System gemeinsam zu bedienen. Dazu werden Aspekte gemeinsamen Musikkonsums vorgestellt, die als Grundlage und Motivation für die Gestaltung der Benutzerschnittstelle in den darauf folgenden Kapiteln dient. Eine abschließende Nutzerstudie soll letztlich zeigen, ob und wie weit die genannten Ziele erreicht wurden.

Abstract

Music plays an important role in the private and social life of today's individuals. This involves in-depth discussions about music and collaborative and exploring searching for music that fits a certain occasion or mood. Conventional systems for maintaining and accessing collections of digital music are inept to support such activities.

Therefore, we present in this thesis the software system “AudioPhield”, which is targeted on tabletop displays. AudioPhield depicts the individual songs from collections of different users simultaneously on a starfield-like user interface. Songs, which are perceived as similar, shall thereby be placed close to each other.

The focus of this work lies on visualizing music libraries and developing suitable forms of interaction to enable multiple users to operate the system simultaneously. Therefore, aspects of social music consumptions are presented, which serve as basis and motivation for the design of the interface in the following chapters. A concluding user study shall reveal if, and to what extent, these goals were reached.

Aufgabenstellung

Currently available software for interacting with digital music libraries does not support casual browsing particularly. Furthermore, these programs are limited to a single user at a time; they lack especially functions to support collaborative, simultaneous browsing.

A system is to be developed that resolves the above outlined deficiencies with a novel user interface targeted on a multi-touch tabletop display. Thus, collaborative, casual browsing of digital music libraries is to be supported.

General solution approach: Songs in music libraries are represented as icons on a two-dimensional surface. The icon layout visualizes similarity relations between songs.

The task consists of three core challenges:

1. Similarity data needs to be extracted from the music.
2. Visualizations for music libraries need to be designed.
3. Schemes for collaborative interaction on a tabletop device need to be elaborated. These should make extensive use of the capabilities of the multi-touch device.

The student has to find solutions to these problems. The found solutions must be integrated into a single, consistent design of a user interface. This interface is to be implemented in form of a fully functional prototype. The student is also required to conduct a user study to evaluate the validity of the found solution.

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig angefertigt, alle Zitate als solche kenntlich gemacht sowie alle benutzten Quellen und Hilfsmittel angegeben habe.

München, 16. September 2008

.....

Contents

1	Introduction	1
1.1	Tabletop displays: Hardware designed to aid collaboration	2
1.2	AudioPhield: A preview	3
2	Related work	5
2.1	Information Visualization	5
2.2	Similarity-based Music Browsing	8
2.3	Social Media Browsing on Tabletop Displays	11
3	Aspects of Music Consumption	13
3.1	Personal Music Libraries	13
3.1.1	Typical Organizations	13
3.1.2	Accessing the Database	14
3.1.3	Finding New Music	15
3.2	Social Aspects of Music Consumption	15
3.2.1	Identity in Private Libraries	16
3.2.2	Collaborative Choice of Music	16
3.2.3	“Music Talk”	16
3.2.4	Music Sharing	18
4	Goals	19
4.1	Usage Scenario	19
4.2	Main Issues	19
5	Interface Design	21
5.1	Relate Pieces of Music	21
5.1.1	MIR-Algorithms for Automatic Similarity Computation	21
5.1.2	Similarity Visualization	21
5.1.3	SOM instead of starfield placement	26
5.2	Visualize Music Libraries	28
5.2.1	Focus and Context: Fisheye Views	28
5.2.2	Information Encoded in the Icons	32
5.3	Enable Simultaneous Interaction	35
5.3.1	Manipulations of Focus Areas	37
5.3.2	Music Playback Control	43
6	Implementation	49
6.1	Hardware Setup	49
6.2	General Software Design	50
6.2.1	Used technologies	51
6.2.2	General Architecture	52
6.2.3	The DataHandler	56
6.3	Selected Implementation Aspects	56
6.3.1	Music Information Retrieval	56
6.3.2	The SOM	64
6.3.3	Spring-algorithm	67
6.3.4	Free-form Fisheyes	69

7	Evaluation	75
7.1	Aim and Expectations	75
7.2	General Study Design	76
7.3	Tasks	78
7.3.1	Single User Tasks	79
7.3.2	Pair Tasks	80
7.3.3	Questionnaire	81
7.4	Results	81
7.4.1	Task Completion	81
7.4.2	General Observations	86
7.5	Identified Issues	87
7.6	Discussion	88
8	Conclusion	91
8.1	Summarization	91
8.2	Future Work	92

1 Introduction

Music is an integral part of our everyday life. Many of our daily activities are accompanied by some sort of music: We wake up to an (hopefully) pleasant tune from the radio clock and listen to music from the car stereo or iPod during transport to work. If we enter a shopping mall, we are exposed to different kinds of ambient music whenever we enter a shop. Music makes us work harder during exercise sessions or helps us to relax after a long day as live music in a pub or a classical concert from the hi-fi system in the living room. And to close the circle, with lullabies there is even music made for going to bed. For many people, this mounts up to over six hours a day [68]!

Why this omnipresence? It is not just that music consists of “special kinds of sounds that tickle [the human sound-processing capabilities] in interesting ways” as described in [74, page 14], music is directly linked to our emotions¹. So, we choose music to match our current mood. But music is also capable of *altering* our mood. Shops utilize this connection by playing music that is supposed to uplift the spirits of their shoppers and, ultimately, encourage them to buy more. However, we use this connection frequently ourselves, for instance if we create playlists for a party or a romantic evening. Therefore, it is important that we choose the “right” music for the occasion - working out to restrained lounge music seems just impossible.

Considering this emotional power, it is no surprise that music has also some very important cultural and social aspects: We define who we are and who belongs to us by music (of course not exclusively, but among other factors like language or clothing). This reaches from national anthems to the music styles specific to a youth subculture (e.g., hip-hop, punk) to the song a couple chose as “our song” (which is often the song, they first performed another activity based on music to: dance). Based on one’s cultural heritage there are also occasions which outright demand certain music: For example, to many people in the Western culture, a wedding is not really a complete wedding without the wedding march. But already listening to music together and rummaging in each others music collection is a deeply social and socializing activity.

Considering that a growing part of personal music libraries is nowadays stored in digital file formats like the ever popular “mp3” or Apple’s “aac”, common music organization and playback software as “WinAmp” ([@23])² or “iTunes” ([@2]) offers surprisingly little support for the above mentioned tasks, namely the casual, non-targeted browsing of entire collections, possibly together with other people. They traditionally offer just long³ lists in which every song is represented as a line containing artist, title and album information. Even with the addition of features as ratings or logging how often or recent a song was played, these systems can not assist in queries of the type “Find me a work-out song” or “Give me a good overview of my friend’s collection”. Especially the social aspect seems to find only attention if the users are at least in different rooms.

This may be due to the computer systems they were developed for: Desktop PCs. This hardware is designed to let one person effectively interact, but performs poor when it comes to multiple users in the same place at the same time. The reason for that is, on the one hand, that only one person can interact with the system at a time because there is usually only one set of interaction devices, such as mouses or keyboards. On the other

¹This relation is arguably known to mankind since the Stone Age and was already studied soundly by Hevner in 1936, see [34]

²Sources marked as [**] refer to content accessed via the Internet. They are enlisted in the “Web References” section at the end of this document.

³Since it is not unusual for private music collections to contain more than 5,000 songs, these lists can even become unmanageable long.

hand, the usual setup forces the users with a vertically mounted, rather small display to sit side-by-side; this design handicaps natural interaction and conversation, which commonly takes place in a face-to-face setting [76].

1.1 Tabletop displays: Hardware designed to aid collaboration

With the arrival of direct-touch tabletop displays, there now exists a new platform that allows users to be located face-to-face while operating the system. Thus, it is especially suited to support collaboration and social interaction of multiple users better than any other known computing environment [69]. The following section shall outline the development and major systems of this technology.

One of the earliest approaches for a tabletop device is the “Video Desk” by Krueger, which was introduced already in 1983. This system was vision based and enabled thus interaction methods that resemble today’s direct-touch approaches – albeit there was no actual touch recognition ([48]). Later systems in the beginning of the 1990s used primarily projectors to create the tabletop displays but still relied on vision-based tracking of fingers or hands for input recognition. Important examples of systems belonging to this category are the “Digital Desk” by Wellner ([93]), or the “Everywhere Displays Projector” developed at IBM research ([26]). These systems already include special modes for multi-user interaction. Later, new technologies to recognize user input were developed. The “DiamondTable” from Dietz and Leigh ([25]) introduced the usage of capacitive sensors for this purpose. This technology not only enables multi-point tracking but also relates input points and users reliably – at least, as long as the latter stay seated and thus connected to a receiver. A very similar approach can be found in the “SmartSkin” system by Rekimoto. In contrast to the DiamondTable, here it is impossible to identify the currently interacting user, and users are not required to stay connected to a receiver ([67]). Paradiso found an altogether different approach in [64]: Here, microphones mounted in the corners of a projection screen are used to triangulate knock-interactions. This technique allows precise identification of different tapping methods (e.g., fingertips sound different from fingernails); on the other hand, it offers no way to identify dragging interactions. Some systems utilizing optical sensors also use triangulation: Therefore, four or more cameras are mounted in the corners of the display so that they cover the complete interaction area from the side. The commercially successful “SMART Boards” [28], e.g., use this technology. The maximum number of simultaneously identifiable depends on the number of used cameras but is generally quite low. Another vision-based approach, which utilizes “Frustrated Total Internal Reflection”, was introduced by Han in [32]. The target hardware for the system developed in this thesis also depends on this technology (see chapter 6.1 for a detailed introduction). All of these technologies were used from the beginning to support social, collaborative tasks. The upcoming “Surface” from Microsoft, which also features a horizontal direct-touch display, seems to aim primarily on social applications, too. With this, tabletop computing seems on the verge of entering the main-stream market ([12]).

So, an interactive tabletop display is a good basis for a system that attempts to enable social and collaborative access of digital music collections – and this is exactly what we want to create in this thesis: A system to support casual, collaborative browsing of music collections. The next section will demonstrate how such a system may look by giving a short preview of the result of our development process.

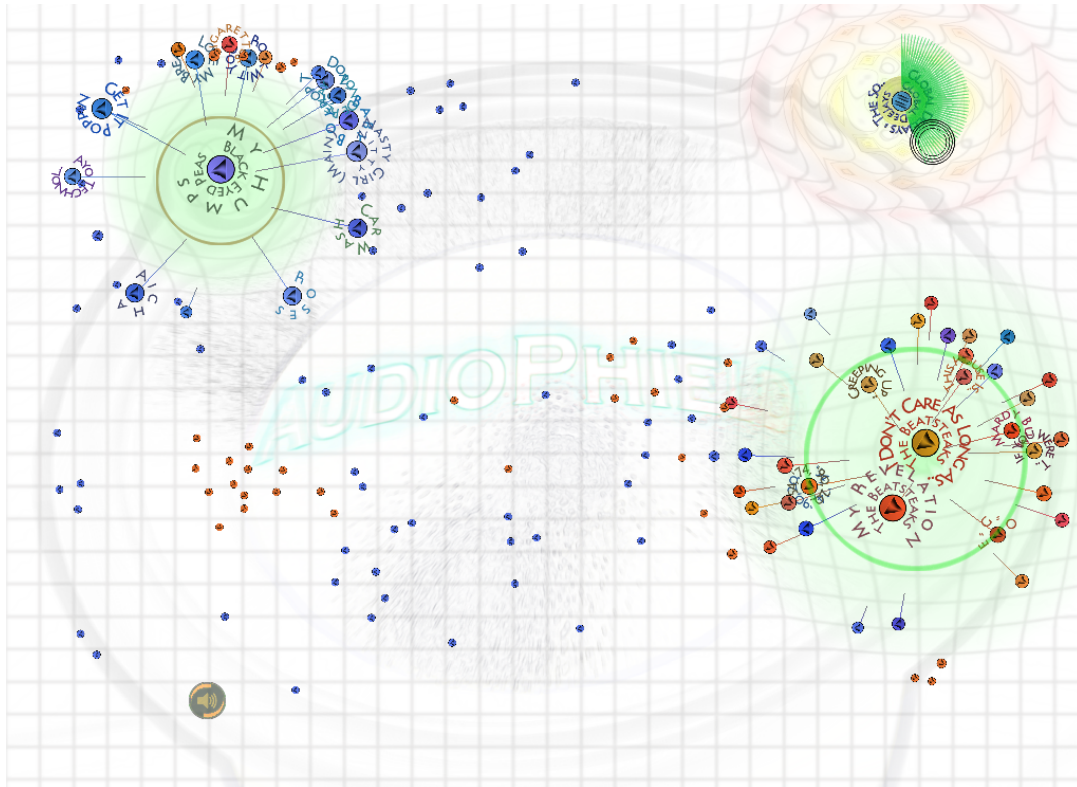


Figure 1.1: In this AudioPhield screenshot, taken from the user study (see chapter 7 on page 75), two people are simultaneously browsing their combined collections. Note the two magnified areas, and the currently playing song in the the upper right corner.

1.2 AudioPhield: A preview

AudioPhield represents all songs in a private music library as small icons spread over the whole display. The placement of these icons follows automatically computed similarity-measurements, which are derived from MIR-algorithms and tagging-data from a social music-service called “last.fm” [7]. So, songs that are perceived as similar are placed close to each other. To obtain more detailed information about an area, users can create and move focus areas. These areas use distortions and magnifications similar to fisheye lenses. Because there is no icon occluded by this transformation, context-information is preserved. There are two ways to manipulate focus areas: ZoomFrames and SoapSpots (used in the screenshot). Songs are either played fully by double-tapping them or just partly for scan-play. The playback volume is controlled by a mobile widget. See Figure 1.1 for a screenshot of the prototype presented in this thesis.

We discuss why we implemented AudioPhield in this way in the following chapters: First, we outline related work in chapter 2. Then we will give a detailed introduction of aspects of typical music consumption behavior, which motivated many design decisions during the development of AudioPhield. Chapter 4 summarizes the concrete goals and tasks following from these considerations. After that, we will reconstruct the designing process of several major aspects of the interface by identifying important subproblems and discussing alternative solutions. Chapter 6 contains implementation aspects as the general architecture and data flow, the realization of the distortions, or the mechanics behind the similarity-based placement. Afterwards, we detail the conduction and results of a first user study in chapter 7 before we finally conclude this thesis with a summarization of the findings and a recommendation of directions for future work in chapter 8.

2 Related work

The task of creating a software system that supports collaborative and explorative browsing of music collections touches different branches of research:

It is foremost an information visualization problem to depict an entire music collection⁴. Then, to make this visualization adept for explorative browsing, we deemed it sensible to base it on similarity data (see section 5.1 for an discussion of this decision); therefore, AudioPhield is closely related to some contributions in the field of Music Information Retrieval (short: “MIR”): Finding similarities between songs and using this information to navigate through music collections is one of the core topics of this discipline. Furthermore, AudioPhield is supposed to aid and encourage collaborative browsing on tabletops.

This chapter is dedicated to outline and summarize some of the major contributions done in these three areas of research. However, we do not attempt to deliver a complete list.

2.1 Information Visualization

The discipline of Information Visualization (short: “InfoViz”) is dedicated to “the use of computer-supported, interactive, visual representations of abstract data to amplify cognition”[12]. So, essentially most of the design decisions made for AudioPhield are related to knowledge from the research discipline information visualization in some way. However, to prevent this chapter to go beyond the scope of this thesis, we mention only techniques to make large collections of data - such as music libraries - comprehensible; other usages of information visualization knowledge for AudioPhield are mentioned there (see especially chapter 5.1.2)).

Making large data collections accessible and comprehensible is one of the core problems of InfoViz. One of the easiest ways is the so called “scatterplot”. It is used to spread a big dataset consisting of at least two-dimensional entries out on a two-dimensional field. Therefore, two attributes of the dataset are chosen and interpreted as Cartesian coordinates⁵ and every item of the dataset is depicted as a point in this space. Scatterplots essentially utilize one of the in 1912 by Koffka *et al.* introduced and 1935 in [45] reprinted *Gestalt laws*, namely the proximity law that states that objects located close to each other are perceived as similar. Thus, scatterplots are a very powerful way to visualize relations between data objects.

While scatterplots already prove to be valuable in statistical software, they become the basis for complete user interfaces through the addition of some sort of interaction. Ahlberg *et al.* were the first who introduced this kind of interface in [2], and they were also the ones who coined the phrase “starfield” for them. These authors also pointed out in the same publication how valuable starfield displays are for explorative tasks. To prove their concept, Ahlberg presented together with Shneiderman, one of the co-authors of [2], the “FilmFinder” in [1] (see Figure 2.1): A complete database of movies is depicted in typical scatterplot-style as tiny rectangles localized according to “popularity” and “year of production”. Users can then change the amount of shown movies on the one hand by applying special criteria (as director, genre, or ratings) through interaction elements placed around the scatterplot. On the other hand, users can zoom into the view to see only a part of the data; which part that is can be adjusted with two sliders placed near the axes.

⁴It seems obvious to *visualize* data to make it comprehensible instead of, e.g., *aurealizing* it. This can be explained by the fact that the perception bandwidth of the human sense of sight is larger by magnitudes than the bandwidth of all other senses [78].

⁵ It is also possible to create 3D-scatterplots by choosing three dimensions as Cartesian coordinates. However, since computer monitors and paper are only able to present two spatial dimensions, these are considered unclear and thus rarely used.

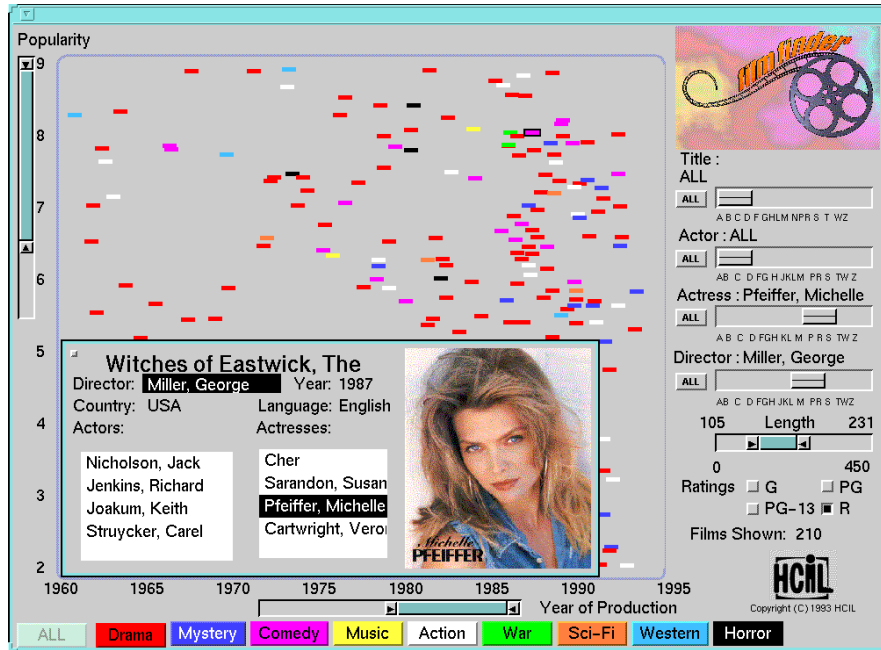


Figure 2.1: Screenshot of the “FilmFinder”. The interface is set to only show movies from 1960 to 1995 with a popularity above 2, and to show details to one specific item. Picture source: [1]

To obtain detailed information about a movie, the user can select one single rectangle to make a detail-window appear.

Ahlberg proved further, how generally applicable and valuable this method of information visualization is, with his company *Spotfire Inc.*: Their products, which are all based on the starfield displays presented above, are used in virtually all industries today [27, 29].

Scatterplots and starfield views have one inherent problem: They are limited to visualizing only two (or in case of 3D-scatterplots: three) dimensions of the data at the same time⁶. When the data is low-dimensional or only few dimensions need to be taken into account at a time, this is not a problem. However, scatterplots can not be applied on high-dimensional data without omitting the majority of the information.

One solution to this problem is to reduce the dimensionality of the data set with mathematical and statistical procedures like the “principal component analysis” introduced by Pearson in [65] or “latent semantic indexing” (see [21]). While these techniques reduce the dimensionality possibly remarkably (dependent on the data), they are rarely able to reduce the dimensionality enough to use scatterplots. And even if they do, the output is usually not suited for information visualization because the reduced set of dimensions consists of *eigenvector*-like factors that correlate to the original dimensions but not necessarily to a graspable concept.

Another possible approach is to compute a “similarity matrix” that holds a proximity value for each element pair in the dataset. Different functions (so called “distance metrics”),

⁶ The number of dimensions visualized can easily be increased by depicting data objects in different colors and shapes as shown in [92, p. 142]. However, here the maximal cardinality of a distinguishable set is rather limited, i.e., not many different values can be encoded in these perception channels to preserve discriminability (e.g., only four different shapes). Also, information presented in color or shape coding are not as immediately comprehensible as spatial relations [55].

like the Euclidean distance⁷ or the Mahalanobis distance⁸ can be used for this computation. This matrix is the basis for various techniques like “multidimensional scaling”(see [6] for an introduction) or “space-filling curves” [71], which try to arrange all the elements of the database in such a way on a plane that the similarity values in the matrix match the (Euclidean) distances in the final layout as close as possible. So, these techniques enable the visualization of high-dimensional data on low dimensional output devices while preserving similarity relations, and are thus nearly as intelligible as scatterplots. They do not create an explicit mapping function, though; that peculiarity requires that the complete procedure needs to be restarted from the beginning when elements are added to the database or taken from it [43, p. 32f]. Also, the resulting layout may look completely different from the original one after such a recomputation. This shortcomings make them difficult to use in interactive systems.

Another possibility to realize similarity-based visualization of high-dimensional data are the “self-organizing maps”⁹ (short: “SOM”) invented by Kohonen. Although they are useful for a wide range of tasks (as, e.g., speech recognition or cloud classification [44]), they have been applied mostly to visualize data since their introduction in [46] from 1981. The SOM is basically a simple artificial neural network that is trained unsupervised, i.e., without correction values or manipulation. The network consists of a grid of neurons, with each holding a vector of the same dimensionality as the input data. The algorithm seeks for each item the neuron whose vector is closest to the input, places the item in the node’s proximity and adjusts the node as well as some surrounding nodes according to the item. A more in-depth description of how a SOM is trained can be found in section 6.3.2 on page 64. After the training process is finished, a similarity-based layout for the input data has been found. Since the influence of new data decreases in most implementations over time, also a stable mapping function evolves from this technique. So, although self-organizing maps do not necessarily deliver the provably optimal layout as the approaches before, they seem better suited for interactive tasks. See [47] for an exhaustive discussion of SOMs and possible variants.

Large data collections – independently of the used layout algorithm – usually can not be depicted completely on the same output device at the same time without massive clutter, which renders the interface unusable. Therefore means must be found to view only clippings of the complete dataset. The usual solution to this problem is zooming and panning – as it was implemented, e.g., in FilmFinder. However, this solution has one big downside: If the magnification is set to a low value, detailed information is missing and the interface seems cluttered; is it set to a high value, then all details are shown but context and overview are lost.

Various so called “focus+context techniques” were developed to enable interfaces that show detail and overview at the same time (see, e.g., [79, 40, 52]; [54] offers a good taxonomy of the general concepts). One of the most popular of these is the “fisheye view”, which was named after the wide-angle fisheye lenses that shows details in the focus area and remote regions in progressively less detail, introduced by Furnas in [31]. This approach computes to every element of the data collection that is to visualize a “degree of interest” (short: “DOI”) value based on the spatial distance to the focal point and the *a priori* importance

⁷ Euclidean distance d_e : $d_e(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$, with $\vec{x}, \vec{y} \in \mathbb{R}^n$

⁸ Mahalanobis distance d_m :

$$d_m(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T \Sigma^{-1} (\vec{x} - \vec{y})}, \text{ with covariance matrix } \Sigma, \text{ and } \vec{x}, \vec{y} \in \mathbb{R}^n$$

See [24] for a complete introduction.

⁹Self-organizing maps are sometimes also called “Kohonen maps” after their inventor Teuvo Kohonen.



Figure 2.2: A graphical fisheye-lens applied to a photo. Picture source: [14]

of the element. The DOI value then determines if an element is even displayed, and if so with what size. Sarkar and Brown extended Furnas’ concept in [73] and made it more generally applicable. This contribution created also the mathematical base for many purely graphical fisheye implementations as the one in [14] illustrated in Figure 2.2. These fisheye views resemble more the projections of real-world fisheye lenses after which the concept was named. However, there exist, to our best knowledge, only a few implemented systems that combine scatterplot-like visualizations with fisheye views. Two examples are the library visualization “star-fish” by Sanchez (see [72]) and a system targeted on PDAs from Büring *et al.* presented in [8], which is supposed to visualize a dataset of books, too.

2.2 Similarity-based Music Browsing

As stated, AudioPhield as an approach to make music databases browsable in an explorative way based on similarity estimations. This is one of the core goals of researchers of “Music Information Retrieval” (short: “MIR”). Insofar, it is no surprise that there are already various attempts to achieve the same. The following paragraphs shall introduce some of them. Most of the systems enlisted below depend to some extent on attributes of the examined pieces of music that are not included in the usual metadata from ID3-tags and suchlike. These values usually get extracted from the music itself by means of various algorithms developed by MIR researcher community. We will not address contributions made in this area of research, but will focus on complete systems intended to browse music only¹⁰.

Torrens: “Visualizing and Exploring Personal Music Libraries” Torrens *et al.* present in [83] a user study in which they compared three different visualizations of a music library. The first visualization they created is a screen-filling disc on which the songs, which are depicted as dots or, if they are part of a playlist, as crosses, are distributed according to their genre and age. The disc is divided into sectors representing all genres in the library. The amount of songs in a genre, in comparison to the whole library, determines

¹⁰Readers, who are interested in psychoacoustics and MIR-algorithms, may find a good overview in [61], [26] and [74], or in our previous work [75]. See also chapter 6.3.1 (p. 57f.) for a discussion of some of the usual techniques.

thereby the size of each sector. The sectors are subdivided in the same manner using artists instead on genres. The outer regions of the disc hold the most recently published songs, pieces of music in the center are the oldest. Users wishing for more detail can zoom into a sector, which is then the basis for a new disc divided into artist-sectors or, in the highest zoom-level, album-sectors. The second visualization uses a rectangle instead of a disc but follows otherwise the same principles and resembles essentially a starfield view. The third visualization uses Tree-Maps: The screen is divided into rectangles representing genres, which are recursively split into rectangles for sub-genres and artists, in such a way that the area of each rectangle depicts the number of songs represented by it in comparison to the whole collection. Individual songs are not displayed. All three visualizations are certainly useful for giving users a good overview of a music collection. However, the system delivers no information of how an individual song sounds (without playing it) or how similar it sounds to other songs – genres are too vague for this.

AudioRadar Hilliges *et al.* introduced in [36] with “AudioRadar” a visualization that is targeted especially on relating songs to each other based on their content, not their meta data. Therefore they compute four high-level attributes like “clean/rough” or “calm/turbulent” for each song. AudioRadar was inspired by real-world radar screens and utilizes this metaphor clearly: In the center of the screen, where in naval situations the ship would be, resides the currently playing song. A few other songs that sound similar to that piece of music are displayed in the surroundings as little “play”-buttons with title- and artist-texts below them. The proximity to the center encodes thereby the general similarity of a song, the sector, in which a song is placed, indicates the dimension in which the song differs most from the central. Users can navigate the “Sea of Sounds” gradually by just double-clicking on a song.

AudioRadar also offers a second interface to create “mood-based playlist[s]”. These playlists are created by specifying value ranges with sliders for all four dimensions of the radar view; only songs with values inside these intervals become part of the playlist. To aid the selection process a typical starfield view is displayed: All songs are placed as dots on a plane. The user can choose which of the four dimensions from the radar view should be used to position the songs.

The consistent radar-metaphor makes AudioRadar easy to understand and offers the user a convenient way to browse casually. Also, since attributes extracted from the music itself are taken, relations are more meaningful than just the usual metadata. However, there are no zooming capabilities or other means to provide users with an overview.

Marsyas3D George Tzanetakis, thanks to his “MARSYAS” framework (see [85]) one of the leading scientists of MIR, presented in [86] a few visualizing modules for his framework. Among them are two interfaces, the “Timbre2D” and the “Timbre3D”, that represent a collection as two- and three-dimensional scatterplots. These viewers are connected to other parts of the framework in a very flexible way. So, any data that the framework can create – including position data from any form of multi-dimensional scaling – could be visualized in one of the spatial dimensions or as color or shape. Intuitive zooming capabilities make browsing the data seem easy; missing labels for the depicted data, however, show that the system is not intended as tool for non-scientific users.

ArtistMap Van Gulik *et al.* present in [90] a browser application that does not operate on song-, but rather on album-level. At first sight, the interface seems to be a ordinary scatterplot with two clear spatial dimensions, like “year” and “tempo”, along which the artists, represented as colored disks, are positioned. Actually, the layout process is more

complicated: For every artist pair a similarity value based on a number of attributes (obtained from webservices or manually assigned) is computed. These values decide if two artists are considered “connected”. The placement is done by a force-directed graph-drawing algorithm. It computes two forces for each artist-disk: A connecting force pulling it to connected disks and a magnetic force pulling it towards its trivial point in the scatterplot. The final layout is the result of multiple iterations of applying these forces until an equilibrium is reached. Thus, a combination of scatterplot-like overview and artist-clusters is displayed.

Search Inside the Music “Search Inside the Music” is a project, which is currently being open sourced ([@5]), for explorative browsing of music collections and automated recommendations developed at the *Sun Labs* of *Sun Microsystems, Inc* under the lead of Paul Lamere. Their user interface consists primarily of three different three-dimensional visualizations shown simultaneously. The first is similar to the Timbre3D browser described above: Colored spheres representing single songs are located according to their musical similarity in 3D space. The position of each sphere is computed from multiple MIR-algorithms combined by a multi-dimensional scaling algorithm. The size of a sphere is proportional to the popularity of a song. When a user clicks on a sphere to play a song, the second visualization, the “Album Cloud” appears. At the center of this cloud, the album artwork of the currently playing song is surrounded by the album covers of similar pieces of music. The third visualization also uses album artwork. It consists of a grid of squares with the album covers as textures, again sorted by similarity. The grid as a whole can take multiple forms, e.g., a loop or spiral around the floating spheres and the Album Cloud [51, 50].

Islands of music Pampalk created in his master thesis ([63]) the visualization system “Islands of Music” (short: “IoM”). The program computes various attributes to each song in the database and trains a self-organizing map with these attributes. This trained map is then the basis for the interface, which follows the metaphor of a geographical map of an archipelago. Pampalk uses therefore a two-dimensional grid in which each square is related to a node in the SOM. Then, the amount of songs assigned to a node is used to calculate a color for the center of each square. By employing a color scale that maps low amounts on blue, medium on yellow, and high amounts on green colors, and by interpolating these colors between the centers of the squares, a visualization emerges that resembles a geographical map of islands. These islands are then either annotated with labels for overview or with a white dot for each song that can be clicked to play it.

Islands of Music was the spark for many similar systems, i.e., systems that utilize a SOM to arrange songs from a collection on a plane¹¹. Knees *et al.*, for example, transformed the IoM in [44] into an actual three-dimensional landscape that can be explored using an ordinary gamepad. Songs are here not played explicitly but automatically if the user comes close enough to their place on the map. Another extension is the “PocketSOM” ([39]) player that targets mobile devices and adds the capability to create playlists by drawing lines on the map. The “MusicMiner” in [58] is also very similar to the original IoM; here, however, the original distances in the high-dimensional space are used for the relief instead of song densities.

AudioPhield as well uses a SOM for the placing process and is in this way closely related to the Islands of Music. However, while the presented systems aim at quite different screen

¹¹However, [63] was not the first publication in which SOMs were used on music collections. This honor belongs, as far as we know, to Feiten and Günzel in [28].

sizes, from mobile phones to wall displays, there is to our knowledge no attempt to create a music browser for tabletops similar to AudioPhield¹².

2.3 Social Media Browsing on Tabletop Displays

The above listed MIR-systems do generally not support multiple users but only one. Thus, while they may support casual browsing, they do not inherently support *collaborative*¹³ casual browsing. This section enlists some approaches for collaborative or social browsing. We limit our considerations on tabletop displays because collaborative systems targeted on vertical displays or remote collaboration have different scopes and requirements [69]. Surprisingly, although there is a lot of research investigating aspects of optimal tabletop interface design like “reach” ([82]), “precise selection” ([5]), or “rotation” ([33]), there are few complete systems.

Personal Digital Historian The early work of Shen *et al.* [77] is one of the first software systems that recognizes the possibilities of tabletop displays for digital media browsing. Targeted on a circular vision-based direct-touch tabletop, the system shows a collection of photos in a radial pattern according to meta data such as “date” or “event”. Users can modify this initial layout simply by dragging individual pictures. The paper also outlines an optional area on the table, which was not yet implemented in [77], where a stream of pictures moves slowly along. Hinrichs *et al.* [37] enhanced this idea in the form of “currents”, deformable display areas, in which pictures float slowly. Users can take photos from these areas and put them back at any time. Thus, currents serve as shared, public repositories.

TVIEWS: Picture Sorter & Map Browser Mazalek *et al.* present in [56] two applications for managing and browsing personal media collections for the “TVIEWS” tabletop display, which uses acoustic triangulation of special “pucks” for interaction. The first application, the “Picture Sorter” is less targeted on browsing than on sorting of photos. For this purpose, the system shows a few photos that are to be sorted into groups by multiple users. The second application is the “Map Browser”, which shows a timeline on a side of the screen from which picture groups can be dragged. Dots on a geographic map in the background depict at the same time where the pictures were taken. The idea of presenting a timeline as a central device to access the photo library can also be found in “PhotoHelix” [35]. Here, however, the timeline takes the form of a helix that can be rotated by means of a special input device at the center of a helix.

The MUSICtable The MUSICtable by Stavness *et al.* [80] fits the criteria of a casual, social browsing system only in the broader sense, but it is one of the few systems concerned with music. It presents a manually created map visualization of a music collection on a tabletop display. This display has no direct-touch capabilities; instead, there are eight buttons mounted around the table. Users can press these buttons to influence the “wind” on the music map, which moves a cursor over the table. The position of the cursor determines the next piece of music to be played when a song finishes. The interface only hints at what music is located where by coloring parts of the map differently according to the genre they represent. The system showed to encourage collaboration but allows only very limited interaction and gives no detail information about the songs.

¹²There is one approach outlined in [38], but to date there is no publication about the actual interface design.

¹³In the sense of a social process of co-located users.

mTable Chiu *et al.* introduce the “mTable” in [15]. Their target hardware is a large screen integrated in a couch table with no touch-input recognition. Multiple users can interact with the system simultaneously with two ordinary gamepads. One of the applications created for the mTable is the visualization of a photo collection in the form of thumbnails scattered over the interface according to similarity. The layout is thereby created by a force-directed algorithm. Users can influence the layout interactively by moving pictures individually or by dragging labels that relate to whole groups of pictures.

Microsoft Surface Microsoft presents in one of the promotional videos¹⁴ for the soon to be published “Microsoft Surface” system an integrated media browser for different kinds of multimedia data. Photos and videos can be placed, rotated and scaled arbitrarily on the interface by touching them directly. Music is represented as album artworks that can be flipped over by a single tap to access the track list. However, precise information of how the interaction is supposed to work in detail and how it performs in supporting social browsing is not available at the present day.

DTLens In [29], Forlines and Chen present their project “DTLens”, which is especially aimed at visualizing spatial data on tabletop displays in a multi-user environment. They use therefore the “DiamondTable” mentioned in the introduction, which provides the application with multi-point touch input. To avoid that users get lost in the data they apply a focus+context-strategy: Simply by pulling two touchpoints apart, areas of heightened magnification are created. These can then be moved or locked for longer examination via buttons aligned at a side of the rectangular focus areas. In a subsequent publication ([70]), Ryall *et al.* studied the experiences of several hundred users, who interacted with the system at different symposia, and conducted that this design found the general approval of the users.

¹⁴ Accessable at [12].

3 Aspects of Music Consumption

AudioPhield is an attempt to create a user interface to assist and support social browsing of music collections. Therefore, we need to know how people usually organize and access their collections on the one hand, and how they talk about, share, and consume music together on the other hand. This chapter illuminates some important aspects regarding these topics for the design of AudioPhield.

3.1 Personal Music Libraries

To understand how to support browsing of a personal music library, we first need to investigate how people organize, access and maintain their libraries. We limit our considerations here on non-professional, casual users. Professional users have a good command of a musical language that is incomprehensible for amateurs, and, most likely, different requirements on music databases[61, p. 28].

3.1.1 Typical Organizations

The organization of personal music collections was usually determined by the dominant physical storage medium. For nearly as long as the record industry – in the modern sense – exists, music was usually obtained in the form of record albums¹⁵, compilations of 30 to 60 minutes of music, regardless if stored on vinyl, tapes or CDs¹⁶. Thus, private collections also feature the album as the basic unit. These albums usually belong to three different parts of the library:

- **The active set** contains the music that is currently most heard. There may be different active sets for different occasions and places (e.g., “car-music”).
- **The main collection** contains all albums that match the current musical preferences of the owner but are less frequently played.
- **The archive** contains music that is stored for nostalgic reasons but very rarely played. This set is not necessarily separated physically from the main collection.

These sets are organized differently: The working set comprehends rarely more than a few albums and follows usually no particular order, while the other sets are typically ordered by date of purchase, date of last playing, major genre, artists, or combinations of these [20]. See Table 3.1 for a summarization.

Similar peculiarities can be found in typical digital music collections, too. Most users maintain a single folder – the analogon to the main collection and the archive – that holds all of their music in the form of a long list of subfolders for each artist or album. In some cases, there is also a (frequently ill-maintained) preselection into genre- or occasion-folders. Active sets can also be found in different forms as special folders or playlists for special occasions or moods. The songs stored in mobile players with small capacities can also be understood as an active set. Organizing the library in the file-system grows less important with the progressive distribution of modern music players like iTunes, which incorporate complex “library”-functions to hide the actual place of storage. These programs allow also accessing the collection by track instead of album – they privilege it even. Furthermore, the

¹⁵“Singles”, i.e., media which typically comprehend just one song in few variations, were in recent years commercially irrelevant[24].

¹⁶To list the most popular. Other formats, like the “DVD-Audio”, which had less distribution, also hold just one album in most cases – in spite of capacities that would allow for more.

	Active Set	Main Collection	Archive
Typical size (albums)	< 10	> 100	–
Music homogeneity	high	low	low
Typical granularity	songs	whole albums	whole albums
Access frequency	regularly	occasionally	rarely
Frequency of change	regularly	occasionally	rarely
Typically sorted by	recency of playback	genre/artist	genre/artist
Typical browsing specificity	low	low / medium	medium / high
Primary browsing information	artist/title	album art	album art

Table 3.1: Major parts and important features of a typical private music library of an adult. A typical size for archived albums is hard to number: It varies too strong depending on age and individual collecting type.

iTunes music store, like all other popular web music stores, sells music also by track instead of whole record albums. Thus, the album, whose popularity arguably stems from the fact that it is the usual form in which music is obtained, might be replaced by the individual song as smallest unit to organize and access private libraries. However, as recent purchase trends [25] show, the “death of the album” as suggested – among others – by Campbell in [11] is arguably at least a few years away. This can be explained to some extent with the notion that the arrangements of albums receive often a lot of attention to relate tracks with each other. So, individual songs are augmented with meaningful context and the album as a whole creates a more intense experience [89, 42].

3.1.2 Accessing the Database

How people access their database depends heavily on their current context. In most cases, in which the listening to music is not their primary activity but rather an accompaniment (according to [68] the more frequent case by large), users will only play songs from their active set. So, they just start the appropriate playlist (or their mobile music player) and listen to albums or single tracks at random. Here, playlists are rather means to define subsets than to play songs in a particular order. The browsing, if there even is any, is thereby usually limited to pressing “next” until a song is found that “fits”. For this, they frequently activate the “shuffle”-function of their player – some users, like the author, even have their *complete* collection on shuffle. Since all of the songs in an active set are well known to the user, the primarily used information for this undirected browsing are the name of the performing artist and the song’s title (see also Table 3.1). Information about the album to which a song belongs is irrelevant to identify a song [7, 9, 20].

When users focus on listening to music, their accessing strategies change: Sometimes, they have a special song or album in mind, which they can easily find by the usually available textual search. Generally, however, their browsing behavior is still rather undirected, albeit their scope is not limited to active sets but may also include the main collection or even archive albums. To support this kind of browsing, other information is necessary: The often found linear search through the long list of all available tracks seems more the

result of the capabilities of current access interfaces than of the users wishes. While typical genres, like the 127 defined in the original ID3-tag format[@22], are rarely used (the artist name seems to convey more information), *idiosyncratic* genres for mood, situation or activity are highly appreciated when available, e.g., in the form of arbitrary tags. Furthermore, browsing for “similar artists” or “similar songs” to a given piece of music is considered very valuable for browsing, and thus it is intensively – and successfully – used through web-services like “Pandora” ([@17]) and “last.fm” ([@7]).

Also, besides textual information such as the artist’s name, album-art is one of the dominant bases for judgment: While it encodes the same information as the combination “artistname + albumname”, it can be faster comprehended. Of course, this holds only true when the user is familiar with the albums. If this is not the case, though, album art might still be valuable since it allows some rudimentary genre estimations. While the mentioned information suffices for browsing the library, it is often not enough to satisfy the users’ wishes for information: They feel their music listening experience is significantly enhanced if they have access to additional data like song lyrics, artist biographies, or video clips. [7, 20, 62, 89]

So, in one sentence, users access their music libraries usually by undirected, casual browsing with the intention to find appropriate music for the current mood or activity.

3.1.3 Finding New Music

There are generally two ways to find new music to add to one’s private library: To go looking for new tunes actively, or to stumble over unfamiliar tracks casually while doing something else.

Actively searching for music meant traditionally going to a store and look for promising albums in the shelves. Since usually nobody has the time to listen into all available records, shoppers typically look for bands they already know or at least have heard of. Or, they try candidates from the current charts. Many modern recommendation systems, from Amazon’s famous “Customers Who Bought This Item Also Bought” to last.fm’s personal suggestions, follow essentially the same scheme (called “collaborative filtering”) of monitoring the behavior of a large group of people to find proximities between different albums. So, essentially, these recommendations can be seen as social similarity estimations [17, 18, @32].

Social recommendations are – between incidentally hearing a new tune en route in the radio, watching TV commercials with an intriguing acoustic background, or live-music encounters in a pub – also the basis for many casual serendipities of new songs: Music is a very popular conversation topic and listening to music together a usual activity. So, it’s only natural among friends to exchange recommendations. Gatherings of this kind are also the usual surroundings when music is shared among friends (see below). Indeed, copying music is sometimes the primary incentive to meet. Personal recommendations tend thereby to be quite accurate, even if recommender and recommendee do not share the same taste [7, 91, 18].

All these different ways to encounter new albums to add to one’s personal library have one thing in common: The new music usually matches the current preferences of the listener. So the new music is most likely very similar to tunes already in the collection.

3.2 Social Aspects of Music Consumption

Listening to music together is in itself a social experience. By creating a sense of common context and inducing similar emotions, music synchronizes the listeners sometimes very bodily at a dance or in surroundings where music is just ambient accompaniment [23].

But, of course, there are more aspects to collaborative music consumption. The following sections will illuminate some of these.

3.2.1 Identity in Private Libraries

Music is directly connected to peoples' emotions and thus to a very private part of their personalities. So, the content of an individual's personal library sheds much light on her character. A related matter is the fact that we define in part who *we*, as a group of people, are, and who belongs to *us* (whatever this group might be) by a common taste of music (among other things like clothing or haircuts). So, music serves as a way to craft and maintain identity as an individual and as a group – and it helps to define the place *inside* the group, too. This manifests itself, e.g., in prejudices: If someone meets somebody who listens to a song the first person likes, too, she will usually feel more sympathetic towards that person than otherwise. And the reverse of this argument holds true no less: Listening to music one deeply dislikes makes the counterpart immediately less likable [7, 17, 57].

Considering these observations, it is not a surprise that most people are hesitant to give complete strangers access to their private libraries. In circumstances where they can expect strangers to see their collection, people often try to manipulate their appearance by presenting their music habits in a specific way. For example, hosts of a dinner party might hide their ordinary pop albums and instead exhume old jazz records to appear more sophisticated; another example is the dismissive utterance “it was a gift” when a forgotten album is held up with a ridiculing glance. This holds even true in circumstances where it seems highly unlikely that people meet in real life: One of the most demanded features of last.fm-users, whose music listening habits are displayed publicly on the service's website, was a function to remove artists from their personal page – and it is now, that it is available, quite popular and replaces practices like mislabeling songs on purpose [9]. *Which* music it is, people like to be seen with, depends entirely on the social context and may vary strongly. Most individuals, however, feel only in the presence of close friends comfortable to present their complete private library [7, 20, 91].

3.2.2 Collaborative Choice of Music

There is a big difference in how people choose music for themselves and how a group of people decides what they want to listen to. This is again closely related to the coupling of musical taste and identity. Most people will hesitate to propose a song, whom they think the majority of the group will not like. Choosing the music single-handedly in a gregarious or even public setting is an expression of power: The choosing person sees herself as the leader of the group and can hence speak and decide for all of them. This behavior can often be seen when the circumstances expose a person; the host of a party or the DJ in a dance club are good examples.

In most cases, however, especially in rather intimate surroundings and peer groups, choosing music is a complicated process which involves various social activities as discussing, bribing, winning & losing, or generally “doing friendship”, as O'Hara *et al.* put it in [60]. As discussed above, this is no surprise: Again, the group is redefining who they are by finding a common taste [57, 60].

The activity of choosing music is generally perceived as well spent leisure time in itself and often interlaced with “Music Talk” as described in the next section.

3.2.3 “Music Talk”

There are, to our best knowledge and according to Brown & Sellen in [7], no studies that focused on how people listen to music together and how they talk about music while doing

so. However, by drawing information from studies like [4] from Bassoli *et al.*, which are foremost concerned with other topics, from our personal experiences, and from studying about 50 threads in the popular music forums¹⁷ “MusicForums.com” [19] and “MySpace - FORUMS” [20], we feel competent to present the following observations and introduce the term “music talk”.

A lot of the conversation around music has to do with the above introduced matters of identity maintaining: The group as a whole defines its identity with crafting a common sense of taste and the individuals define their space in the group by relating to the preferences of peers and their common ground. This kind of conversation needs – and creates – a sense of intimacy between the participants because the revealed preferences are often something highly personal [23, 91].

The playing music itself is often the center of the conversation: The listeners state how they generally feel about a song (which can involve strong feelings: “*I HATE this song [...]*” [19, p. 475]) and how they estimate its general quality (according to user “Rollxy” Billy Ray Cyrus’ “Achy Breaky Heart” is “[...] *the worst song to ever be sung on this earth*”). These statements are frequently refined by remarks about the specific qualities of a song like the general composition (“*the bass and guitar just click right*”) or individual performances. The conversation also includes comparisons with other songs and albums from the same artist (“*Their old stuff is really good, but the recent s***?*”) or different artists altogether (“*he has, like, the same voice as [...]*”).

Very different, but closely linked nonetheless, from the just described music-centered talking is the music-*triggered* talking.

As discussed in the introduction, music has the power to affect someone’s emotions. This association works in both ways: If people experience moments of strong feelings, the music that accompanies that moment becomes an integral part of the memory about this moment – even more so, if the song in question was previously unknown to the person; a couple’s “our song” is a common example. So, when the same song is played afterwards, it conjures up the mental image of the original experience. In the same way, strong emotions installed *by* a piece of music become associated with the current surroundings, activities, and people. The connection varies in strength according to the intensity of the felt emotions: In rare cases a piece of music becomes the “theme song” for an event, but normally the association is fainter [4, 23].

Thus, music can trigger memories, of events actually linked with the music like a live concert, or of events which were accompanied by the given song randomly (e.g., a song playing in the radio on the way to an vacation resort). These memories become part of the conversation and can lead to common reminiscing about the events associated with the song. If not all members of the group share the conjured memory, others usually tell the story. Of course, the initially triggered memory or its generally mood may spark memories – and thus talking – of other, in any way associated events or songs. Those topic-“jumps” occur spontaneously and the linkage may be impossible to anticipate. However, the general topic “music” is usually preserved [4, 23, 89].

All the mentioned aspects, the analyzing of the playing music and the performing artist as well as the reminiscing, are inseparably interwoven and build together what we want to refer to in the future as “music talk” in analogy to the term “photo talk” coined by Frohlich *et al.* in [30]: Music talk is also an unstructured and meandering process of discussing

¹⁷We are aware of the fact that this is written conversation between mostly anonymous users, not actual face-to-face chatting. However, many of the users, who have often already common ground by feeling similar towards the same music, share each others presence in the same threads long enough to have built up intimate relations. This, and the resemblance of casual conversation on most threads make these forms of interaction arguably close enough to ordinary talking to strengthen our argument.

casually the actual documents and reminiscing together or telling the stories of the events associated with a document – only with music instead of photos as focus point. If music talk happens to take place in surroundings, where the participants are in control over the played music, presenting of songs and pointing out peculiarities takes place similar to the behavior observed by Frohlich [60].

3.2.4 Music Sharing

The sharing of music, i.e. giving a friend a copy of tracks from one's own private collection, can be seen as the natural extension of "music talk" as introduced in the previous section. However, if the sharing is perceived as a current task and not the consequence of an incidental finding during general music discussions, the conversation and behavior of the participants changes and becomes more asymmetrical: Usually one participant takes a serving role by browsing her collection to find music she can recommend. This role can get passed on to other members of the group if they have their private music library at least in part with them – with the storage capacities of more than 20 gigabyte found in current mobile music players, this is often the case. If a piece of music is unknown, it tends not to get fully played but just to "the best part" before it is found worthy of being copied or discarded; then it may only continue to play until the next potential recommendation is found. However, the social patterns around the sharing of music are basically the same as usual music talk, just more focused and less symmetrical [7].

It is worth to note that all these observations apply only to face-to-face meetings between friends. The popular – and often condemned – peer-to-peer file sharing networks as "KaZaA" and "Gnutella" lack this kind of complicated social processes; social interactions are here limited to sharing own songs with others because of a vague sense of reciprocity [68, 91].

Many of the enlisted aspects base on the capabilities of common, currently available devices to store and access music. As history shows, music listening behavior may well change when new systems become available. This was the case with the advent of tape recorders, when people started to create their own mixtapes and copy their songs for friends. Another example is the Sony Walkman, which empowered listeners to override the "soundworld" of their surroundings with their own soundtrack. So, it is reasonable to expect that new technologies will also induct new behavior [59].

4 Goals

The last sections elucidated topics and contributions related to the overall goal of creating a system to support casual, face-to-face browsing of music collections on a tabletop display. This section is dedicated to specify and refine this task by deducing from the presented knowledge.

4.1 Usage Scenario

As seen in chapter 3, people’s behavior when listening to music varies strongly depending on their context. Since these differences in behavior make also very different, in parts conflicting, demands, *one* system may arguably not suit all of the possibilities. We limit AudioPhield’s scope therefore to the following usage scenario:

AudioPhield is supposed to be used with a multi-touch tabletop display in private surroundings by few (one to four) concurrent users. These may leave or join the group at any time but they usually participate in the interaction (actively or passively, i.e., mere observing) for a longer period of time. All users are supposed to be interested in music¹⁸ and know each other well enough to feel comfortable to present their private music library (see paragraph 3.2.1 on page 16). Browsing and listening to music is the main occupation of the group. Their browsing is casual, i.e., no specific song is to be found. Furthermore, the users do not primarily aim to copy music from each other.

In this scenario, we can refine the task “support for casual, collaborative browsing” with the previously established terminology: AudioPhield shall support “music talk”, as introduced in 3.2.3, as good as possible. The system should thereby at least enable symmetrical music talk by not forcing any user to take a fixed role (as, e.g., the “presenter” in section 3.2.4).

4.2 Main Issues

A system that performs satisfactorily in the described usage scenario has to feature solutions for the following coherent issues:

Relate Pieces of Music

Casual browsing is to be supported. Therefore, the usual lists sorted lexicographically after artist or genre do not suffice because no specific song is to be found. AudioPhield needs thus other ways, which are adjusted to the actual needs of the user as discussed in the previous chapter, to order and relate items of a private music collection and to visualize these relations.

Visualize Music Libraries

The task “browsing” demands a suitable depiction of private music collections according to Shneiderman’s well-known mantra “overview first, zoom and filter, then details-on-demand” [78]. This includes: What information is presented when in which way? Also, since the system must not prevent symmetrical music talk, it needs to be able to depict multiple libraries at once.

¹⁸This is not a strong constraint: According to [68], 80% of the questioned people (youths up to the age of 24) rated music as an “essential item”. In comparison: Only little more than 60% saw their mobile phone as an “essential item”.

Enable Simultaneous, Collaborative Interaction on a Tabletop Display

Interfaces targeted on direct-touch tabletop displays have to cope with a number of issues unknown in the design of common desktop applications: Since users may sit or stand at any place around the table, there is no obvious orientation for text; also, users may collide physically when changing their position, e.g., to access parts of the interface that were previously out of reach. Furthermore, the target hardware for AudioPhield (see 6.1) makes multi-touch input possible. This kind of interaction offers a lot of potential – but the best way to exploit this potential is still a matter of research.

Of course, these subproblems are not really disjoint and cannot be solved independently. Nevertheless, this segmentation exhibits a good structuring of the design process and will therefore also be used in the next section, where solution approaches and individual design decisions will be discussed.

5 Interface Design

This chapter illustrates the design process of AudioPhield. It consists of a list of subproblems; to each subproblem, we present and discuss our solution and possible alternatives. The succession mirrors roughly the order in which we identified and discussed the individual problems. The scope here is *what* is to be created and *why*. The “*how*” can be found in the chapter “Implementation” afterwards. In reality, of course, the design- and implementation-phases were not as clearly separated as suggested by this dichotomy but rather interleaved.

5.1 Relate Pieces of Music

As explained in chapter 3 (especially 3.1.2 on page 13), the mood of songs and similarities between songs are most important features for casual browsing. Mood can thereby be seen as *included* in similarity estimations: If two songs are perceived as generally similar (in contrast to being similar regarding a special aspect), the mood associated with the songs is usually alike (see section 3.2.3). Thus, for optimal support of casual browsing, AudioPhield needs to be able to order and relate songs according to their perceived similarity.

5.1.1 MIR-Algorithms for Automatic Similarity Computation

In order to employ similarity estimations, means to compute indicators for similarity need to be found first. A good source for these estimations are the various algorithms developed by the MIR research community. Although the task of content-based automatic extraction of perceived similarity is generally considered unsolved, we yet hope to achieve reasonable precision and robustness by implementing some of the most popular techniques. At this point in design-process it is not clear which algorithms will eventually be used; for the selection we plan to pursue a trial-and-error approach in the implementation phase. However, it is already clear that we will use the framework from our previous work (see [75]), which already includes some of the most frequently used algorithms in MIR (e.g., Short Time Fourier Transformations or Constant-Q Transformations) and allows rapid and flexible addition of new ones.

To use content-based extraction techniques for the similarity measurements has a few drawbacks. The gravest is that there is no ground truth. Daniel Ellis *et al.* cast some light on this problem in [27]: First of all, pieces of music may be perceived as similar or different in multiple dimensions, e.g., speed, instrumentation, musical key, or lyrics. Usually, people use a combination of these if they are asked to assign a single likeness-statement for two songs; which combination that is, i.e., how much weight for each dimension is used, can not be deducted from the audio information itself. It depends on personal factors, as preferences and familiarity with the questionable music, and may well change over time. Also, as Pachet *et al.* discuss in [62], some criteria to consider two songs alike may not even be contained in the audio signal: This applies to cultural influences and meta-data (songs of the same artist are generally considered similar) as well as to highly personal experiences (see page 17). However, even if perfect automatic similarity computation is impossible, [27] showed also that there is still enough common ground to expect estimations which seem sensible to most people of the same culture group.

5.1.2 Similarity Visualization

Having similarity information of all items in a personal music library is an important step towards supporting casual browsing, but in itself does not help much. While the usual

list views of common music playback software may surely be enhanced if the user can sort them according to alikeness to a specific song, they still seem inappropriate for the given task: There is no way for the user to judge, how close a song in the list is to the given one (only more or less similar than songs in the list above and below) and how two songs further down in the list relate to each other. Thus, AudioPhield needs to visualize the relations between the songs in the library in a different way. To decide, how to design such a visualization, it is important to know the characteristics of available display dimensions and their combinations. The following excursion summarizes some of these considerations.

Excursion: Information Coding

Information can be encoded in various display dimensions. The most important and frequently used dimensions are spatial position, size, color, shape, connection, orientation, motion, and texture. Most of these variables may be split into sub-dimensions: E.g., the position along the x-axis may encode different information than the y-position, or color may be differentiated according to the used color model into a red, green and blue channel or into hue and saturation (see Table 5.1). While in principle all display dimensions are capable of visualizing any kind of data, they are not equally adept for certain tasks. Their adeptness depends fundamentally on the type of data that is to be encoded: Nominal data demands visual representations that are clearly distinguishable, other data requires that the relationship between two items (ordinal data) – or in the case of quantitative data: between items and a fixed zero-point – is comprehensible depicted. Figure 5.1 illustrates the different applicabilities [92, p.176 ff][55].

Related to the data type requirements is the monotonicity of the visual dimensions. For monotone dimensions it is easy to understand if visualized data items are greater or less than each other. With some tricks it is possible to turn every display dimension monotonic (e.g., while hue is not considered monotonic, a yellow-to-blue transition is); however, some display attributes, like size or y-position, are intuitively understood to encode quantitative values and can thus be seen as “natural monotonic” ([92, p. 182f]). Thus, natural monotonic ways of visualization are best suited for quantitative data.

Especially if nominal data is to be visualized, the concept of “preattentive processing” is important. It means – simplified – that classification tasks are executed “at a glance” without individually judging depicted items. For example, a white pixel in a black rectangle of almost arbitrary size can be found in an instant without evaluating the brightness of every individual pixel. So, preattentive processed encodings are quite valuable for tasks which involve speedy classification. While all presented display dimensions can be processed preattentively, the number of distinguishable discrete values varies strongly. Table 5.1 presents the maximal recommended cardinality of value sets to display for each dimension. These numbers are only valid if only one dimension is used for visualization at a time; the more different dimensions are combined, the less values can be distinguished preattentively [92, p. 149ff].

In most cases, multiple of these dimensions are combined into one visualization to enrich the information density. Thereby, one should not neglect the fact that the different means of coding information are not isolated but may interfere with each other. Some combinations are plainly impossible: E.g., if circles or dots are used as a shape, orientation loses its meaning. Other combinations are not mutually exclusive but should still not be used together as they tend to be hard to read. Table 5.2 illustrates the separability of some visualization-pairs. Generally, the readability of the visualization decreases with every more

¹⁹Connection is essentially binary: Items may only be connected or not connected.

²⁰The number of distinguishable values regarding one icon pair is 1 (see previous footnote ¹⁹). However, connections between virtually infinite pairs of symbols are still immediately perceivable.

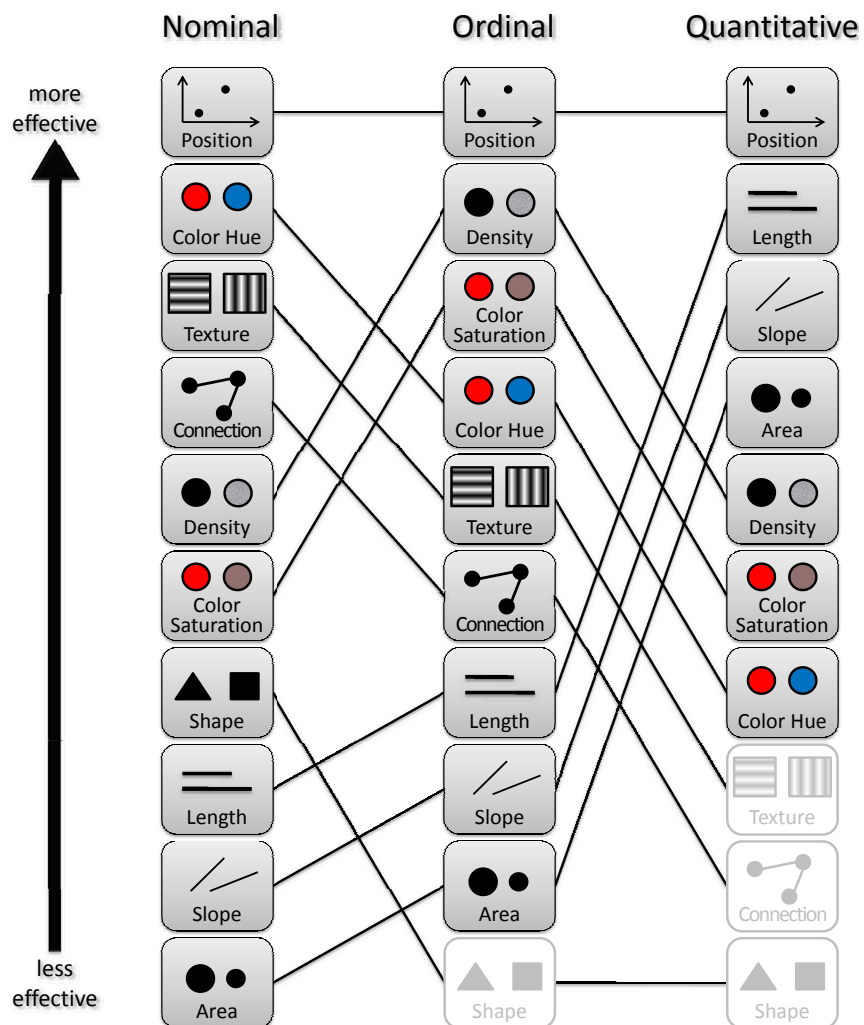


Figure 5.1: Effectivity-ranking for various display dimensions on the basis of [55] and [16]. Dimensions depicted in gray are invalid for the given data type.

Visual variable	Dimensionality	Natural Monotonicity	Preattentively Distinguishable Steps
Spatial Position	2 (X, Y)	Yes	>15
Color: Hue / Saturation	2	Yes (Saturation)	8
Color: red-green / yellow-blue	2	Yes	8
Size	2 (X, Y)	Yes	4
Orientation	1	No	4
Texture	3 (orientation, size, contrast)	No	3
Shape	1	No	4
Connection	1 ¹⁹	No	∞ ²⁰
Motion Coding	2-3	No	2

Table 5.1: Characteristics of various InfoViz channels after [92, p. 182f]. Regarding shape: If different aspects of a shape are changed, more than one dimension is possible as demonstrated with the so-called “Chernoff faces” in [92, p. 239]. In most cases, however, these combinations offer poor readability.

used dimension. Therefore, a tradeoff between information density and readability needs to be found.

All of the above observations and considerations must be seen as strongly simplified: There are a number of ways to encode data and influence factors on sensible selection of display dimensions for information visualization not discussed here, and unusual cases, where the presented recommendations are wrong, may easily be constructed (e.g., the color of very small shapes may be hard to recognize). An in-depth examination of these topics can be found in the chapters three to six in [92]. The explanations of this excursion, nevertheless, should suffice to provide the foundation for many of the following design decisions.

As stated before the excursion, similarity is the most important relationship in a music library for casual browsing. Thus, it should be visualized through spatial positioning, the most effective way to visualize data (see Figure 5.1). Also, as a further argument, spatial proximity is intuitively understood as similarity (see section 2.1). The straightforward approach, which we also initially employed, is to use a starfield view²¹ as introduced in section 2.1 and illustrated in Figure 5.2(a).

Here, the question arises, *what* is to be represented as icons on the starfield. There are three possibilities: Artists, albums or individual songs. It is not uncommon to bundle very different tracks to one album. Depicting albums as small icons at a specific point on the

²¹From this approach stems the project’s name: AudioPhield is the playful combination of “audio” and the fusion of “-phil” (Greek: loving) and “field” from starfield.

	position	size	shape	color	x/y motion	rotation
position		++	++	++	--	+
size	++		-	o	o	-
shape	++	-		+	+	-
color	++	o	+		++	++
x/y motion	--	o	+	++		o
rotation	+	-	-	++	o	

Table 5.2: Separability of some display dimension pairs. Entries range from ‘++’=“very good separable” to ‘--’=“strong interferences”. The table is based on [92, p. 180].

starfield fails at incorporating this bandwidth; also, if the average attributes of the songs of an album are taken, the result may be very misleading: If a pop album contains about an equal amount of two types of songs, slow ballads and fast, upbeat dance tracks, then the album would be localized among average fast/slow albums – although there is no track in the album suiting this spot. Of course, it would be possible to stretch the icon in such a way that it covers the attributes of all songs. While we have not tested this approach, it seemed to us that the resulting overlaps would render the visualization very cluttered and unreadable. Another problem is that people associate usually just a few songs with an album; in some cases, these songs differ strongly from the remaining album. Thus, even if the album is mostly homogeneous, it would still be localized at an unexpected and seemingly wrong spot (if, e.g., a listener’s association with Evanescence’s “Fallen” is the acoustic “My Immortal”, she would be surprised to find it in an area of fairly hard gothic rock). Furthermore, if only albums are depicted, the need for an extra visualization of the not shown contents behind an icon arises. Such detail-windows, as implemented, e.g., in the FilmFinder (see Figure 2.1), would reduce the available space for the field – especially since there should be at least one such frame for each user to avoid interferences. Worse, it would make the interface more complex and thus violate the time-proven “KISS” principle: “Keep it simple, stupid”²². All these considerations apply for depicting artists as icons even more since here the bandwidth of different songs is even wider. Thus, AudioPhield should visualize individual songs as icons instead of albums or artists.

As a drawback of this decision, the number of icons on the starfield view will become very large – since typical private music libraries contain at least 3000 songs. Also, the field does not implicitly display a track’s affiliation to an album; this information, however, supports casual browsing (see 3.1.1, page 14) and music talk, and needs therefore to be visualized in another way. We propose for this purpose to link songs from the same album visually by drawing lines between them. This method is chosen because a high number of necessarily distinguishable values (each album represents here one value) needs to be visualized; as Table 5.1 shows, connecting lines are well suited for this task. Of course, visualizing the album belongings of all songs at the same time would lead to massive clutter²³, so the album information should only be presented on demand.

²²The original creator of this acronym is unknown. However, it was already considered well-known in Lampson’s influential publication [53] from 1983

²³For a collection with a albums and s songs per album on average, $a * \sum_{i=1}^{s-1} i$ lines must be displayed. This would be 6600 for a common library of 100 albums and 12 songs per album.

5.1.3 SOM instead of starfield placement

As a drawback for the starfield approach, it became clear that there are no two axes to align all songs in the collection to in such a way that songs which are perceived as similar are located close to each other – there are just too many dimensions in which pieces of music can be alike or different. As a solution, we pursued the idea to let the user choose the axes for herself from a small, high-level set of musical attributes. Thus, even if general similarity is still not depicted, at least this visualization allows to specify in which regard songs should be alike to be located together. To extract a set of attributes from the songs that is at the same time complete (i.e., sufficient to at least satisfy the most common sort-tasks), meaningful (the user has to understand the criteria) and possible to compute could have been a complex – if not impossible – challenge. However, the approach is unsuitable for AudioPhield because of another reason: Changing the axes causes a complete reorganization of the entire interface. So, if more than user is interacting at a time (what is definitely to be supported by AudioPhield), the browsing activity of all users is suddenly interrupted, and they have to reorient themselves. As a consequence, the axes are either very rarely changed to not annoy anyone (what renders the possibility to do so worthless) or only one person can interact at a time – in this case, the application is effectively useless for collaborative interaction. We will refer to operations of this kind, which must generally be avoided in our task, as “*disruptive*” in the following considerations.

So, while the strength of starfield views, namely using spatial encoding for similarity, should be preserved, the simple layout technique can not be used. As already discussed on page 6f, mathematical means to reduce the dimensionality of the input data is not usable here because pieces of music differ in just too many ways; if these are combined to only two dimensions, the result is arguably close to meaningless. Also, techniques based on similarity matrices like “space-filling curves” (see [71]) are unfeasible: The locations of each song is unpredictable and may be completely different when the database is changed. While we expect that additions or removals of songs occur rarely, it will nevertheless happen, e.g., when a new user joins the group and adds her music library to the interface. Strongly varying layouts make it hard for the users to navigate their collections.

Therefore, we decided to use an approach based on a self-organizing map (see 2.1) to place the song icons: When a SOM is trained, the algorithm builds up an explicit mapping function. Thus, when new songs are added to the database, no recomputation is necessary and the location of all previous icons remains the same. The visualization created by the SOM is solely based on similarity measurements and offers no explicit axes. That is to say that the similarity between songs reveals itself instantly and should therefore support casual browsing, which necessitates similarity estimations (see 3.1.2), considerably better than a starfield view. Figure 5.2 illustrates the differences.

On the other hand, the placement is not as transparent as the straightforward approach of the starfield. So, when a user browses the interface, it is not a priori clear to her in which way the music will change if she moves in a certain direction. To reduce this problem, we devised some forms of additional navigation aids, as letting the user mark some well-known songs to turn them into “beacon song”, which would have special graphics (e.g., a halo) around them, or just “beacon points” which would augment some points on the field with descriptive information in form of texts (e.g., “hard rock”) or symbols (e.g., a guitar). Experiences with prototypes showed that actually the mental map of the field, which is obtained quickly after a few minutes of browsing, suffices to navigate the library. So, navigation aids were removed from AudioPhield’s design because the extra display space they would occupy seems not justified by the benefit of additional orientation limited to the first experiences with the system.

A SOM is usually initialized with random numbers. It is thus not possible to predict

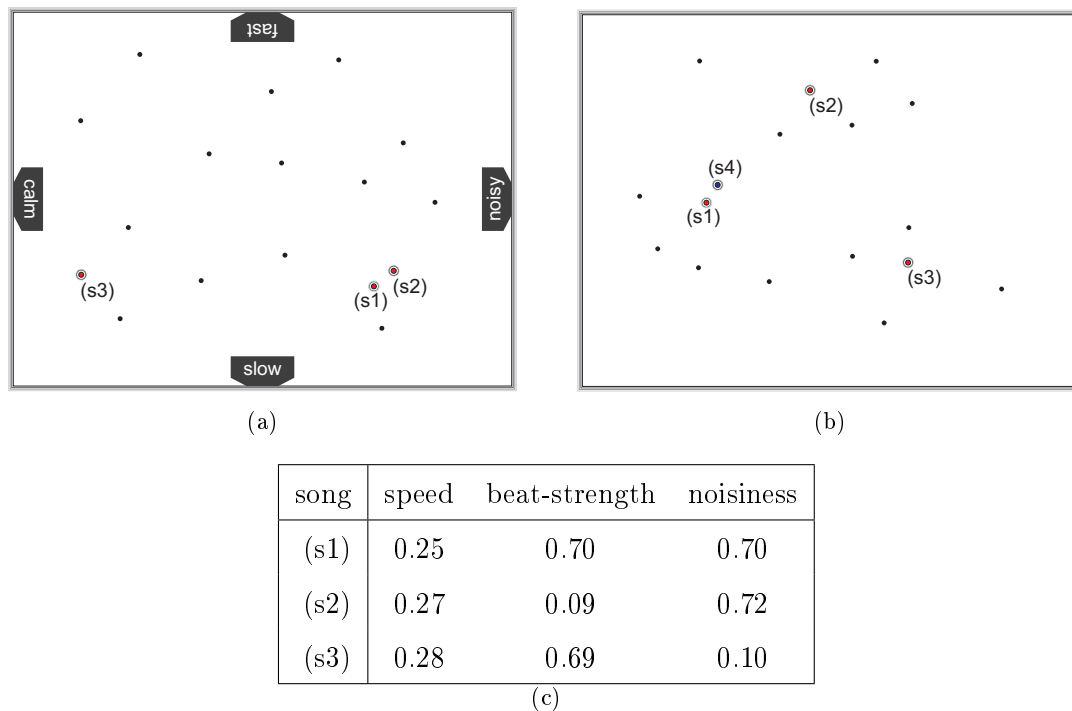


Figure 5.2: Illustration of the different localizations of starfields and SOMs.

(a) shows a starfield view of an fictive music collection with “noisiness”(X) and “speed” (Y) used for positioning. Note, how the songs s1 and s1 are closed very close together and thus expected to be similar, although they differ strongly regarding “beat-strength” (see c)).

(b) shows a layout of the same collection as in a) created by a SOM. Here, the distances between(s1), (s2) and (s3) allow immediate judgment of their similarity. Also, very similar songs may be instantly found, e.g. (s1) and (s4).

(c) contains the attributes of three arbitrary illustrative songs.

where which type of music will be located. So, the same problem as above arises because the layout of the visualization changes after each restart of the application – albeit no more after it was started. Yet, this problem can be minimized by pre-imprinting the SOM. This imprinting should be globally defined, i.e., it should be identical for all installations of AudioPhield. So, while the localizing process creates still individual results for every music library, the general layout should always be about the same. In this way it should be ensured that multiple users can still navigate the interface easily when the system displays their libraries together. Otherwise, if one user inserts her collection into the visualization of another user, the result may be at best unfamiliar and in the worst case unusable: Consider, e.g., a user who listens exclusively to classical music and has therefore chosen to imprint her SOM in such a way that classical music spans over the entire display; if another library containing, say, mostly pop songs is to be placed by the same SOM, it would get crammed into a single small spot. On the other hand, if the pre-imprinting is too dominating, the SOM can no longer adjust to the peculiarities of a collection and valuable display space would be wasted. So, a tradeoff needs to be found.

5.2 Visualize Music Libraries

The previous section exposed that music libraries should be depicted as song icons located according to their similarity on a plane. While this outlines the general direction to go, the interface is still far from defined. This section will introduce and discuss how AudioPhield should look.

5.2.1 Focus and Context: Fisheye Views

Since the music libraries are to be visualized as one icon per song, the visualization will easily contain several hundred icons. It is impossible for this amount of items to display identifying information, i.e., at least artist name and title (see 3.1.2), for each song because there is just not enough display space. Especially considering the fairly low resolution of the target hardware (see section 6.1 on page 49), display space is a precious resource. The usual solution for this problem in starfield-like views is to let the user browse the database by providing her with means to zoom into the data and scroll the current cutout. Thereby the number of details shown for each icon increases analog to the magnification. This is possible because the spreading of the icons at higher zoom levels frees the necessary display space. In this way, these “pan&zoom interfaces”, as they are often called, conform to Shneiderman’s fundamental mantra of information visualization: “Overview first, zoom and filter, then details-on-demand” [78]. Also, users are already familiar with this kind of interaction as similar operations are to be performed in standard desktop software from wordprocessors to image editors. However, this interfaces also exhibit some major flaws that render them inappropriate for AudioPhield. First of all, they only offer overview *or* details but not at the same time. So, when the interface is set to a magnification that allows to identify individual icons, the immediate surroundings are not shown – let alone possibly connected but remote items. Users can therefore hardly relate the currently depicted data to the unshown remainder. Also, users sometimes get “lost” and need to zoom out completely to reorient themselves. While these drawbacks might still be tolerable, the following one is not: Pan&zoom interfaces are “disruptive” (as introduced above) because every zooming or scrolling operation inevitably interrupts the interactions of another user.

So, another solution is to be found. We examined therefore so called “focus+context” techniques, especially graphical fisheye views because they seem best suited for planar information visualizations. As the name suggests focus+context presentation techniques are supposed to ensure that details in a focus region are perceivable while context information

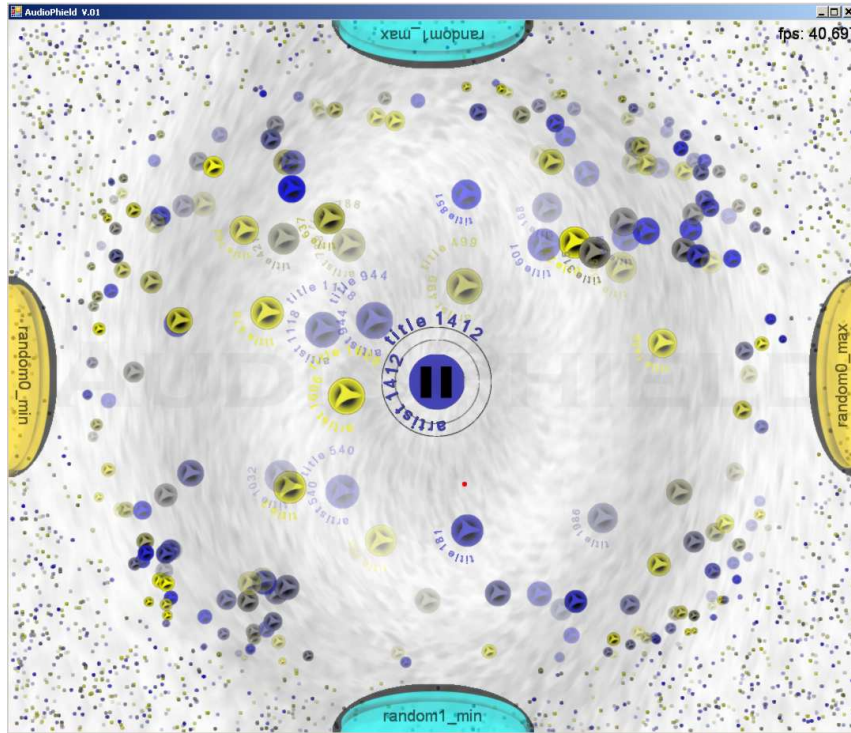


Figure 5.3: Screenshot of a prototype with a static, central fisheye lens. Depicted is a test-dataset of random data.

is still available. Graphical fisheye views attempt to achieve this by a distorting transformation of the planar source data with a smoothly modified magnification factor. Figure 2.2 in section 2.1 on page 7 illustrates the effect on a picture. Our first attempt with this technique was a static fisheye lens in the center of the screen, which affected the whole screen. In contrast to a simple graphical fisheye, the distortion modified here the position, size and amount of shown details for every icon, but not their shape. The database was moved under this lens to browse the music library. See Figure 5.3 for a screenshot of a prototype featuring this zoom lens.

The minor problems (for our task) of pan&zoom interfaces are hereby solved or at least strongly diminished: Detail information is still displayed – and at the same time the relation of an item to the immediate surroundings and the collection at large. Also, the probability of users getting lost in the data is rather small since the system allows permanent overview and the magnification occurs not in discrete steps but smoothly when the field is moved under the lens. Yet, this approach is still disruptive: Since there is only one focus region, it is impossible for two users to access different parts of the depicted database. A possible solution would be to use one static fisheye for each user and duplicate the library to allow them to browse independently. The system would therefore split the available display space into two (or more) separate regions. We dismissed this approach in part because of the scarce resolution of the target hardware, which would make it virtually impossible to show details, such as labels for a song title, and a reasonable amount of context without massive overlappings and clutter. Furthermore, it would equal a setup where ordinary PCs are just located near each other, and the social and collaborative possibilities of the tabletop display would remain unexploited.

The next, and for now final, approach uses multiple, small fisheye views on the same dataset. Here, instead of moving the library under the lens for browsing, the fisheye lenses are dragged over the field. Thus, the whole collection needs to be visualized at the same

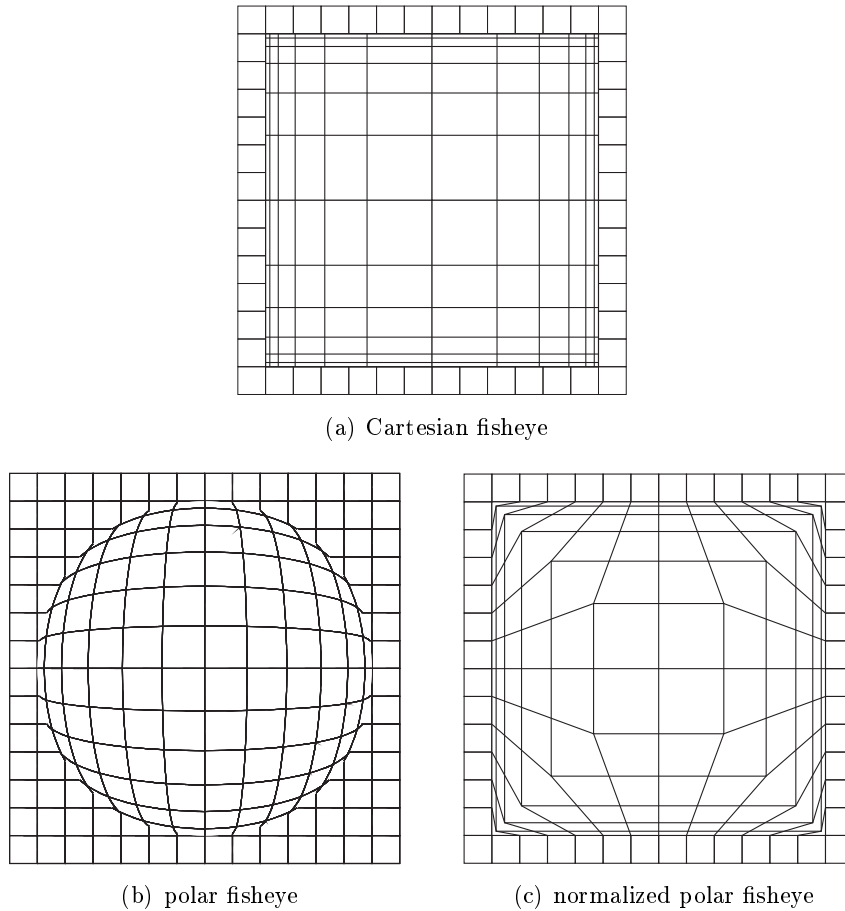


Figure 5.4: Examples of different graphical fisheye distortions

time because any form of utilizing means to show only a part renders the interface as disruptive as the pan&zoom approaches. In this case, all needs should be satisfied: The user interface follows Shneiderman’s principle and offers overview, zooming and details-on-demand²⁴. Also, it is non-disruptive since there is no operation involved that changes the layout of the whole visualization. Users may still interfere with each other, though, e.g., when fisheye lenses overlap each other. This kind of meddling is foreseeable in a collaborative environment and can hardly be avoided. We expect users to develop social protocols to minimize those disturbances – as was observed in several other tabletop environments (see, e.g., [84, 81]).

The question, which *kind* of fisheye view AudioPhield should feature remains to be answered. The choice between a Cartesian and a polar fisheye (see Figure 5.4) is already determined by the fact that the lenses should only magnify a small region of the display and needs thus to create a fluid transition between magnified and normal areas. This is only possible with polar fisheyes because Cartesian lenses, which apply the magnification for each space-dimension separately, have abrupt transitions at the edges of the lens (see Figure 5.4(a)). The Cartesian fisheye distortion had one advantage, though: If the lens is dragged over an area of icons, their directional relation is preserved: Icons with the same x- or y-coordinate are clearly placed on the same line, regardless of their magnification. In the case of the polar fisheye distortion, icons follow a curved path when the lens passes them (see Figure 5.4(b)). Thus, the directional relation becomes hard to estimate. This flaw might be significant in starfield visualizations, but it is arguably negligible for AudioPhield because,

²⁴see, e.g., the album connecting lines below

with the SOM-placement, the x- and y-axes have no clear meaning; so, little information is obscured. That is not to say that directions had no meaning on AudioPhield! But these directions are rather meaningful in terms of larger distances (e.g., a song is part of a rock area but also close to slow pop songs) than in close proximities.

Another choice to be made is if the fisheye distortion should be normalized (see Figure 5.4(c)) as recommended by Sarkar and Brown in [73]. We decided to utilize the not normalized, circular version in part because it integrates better with the remainder of the interface, where circular shapes dominate (see below). Furthermore, this type resembles more a real lens and should so appear more familiar to users.

However, while we approve the general concept of a fisheye distortion, we do not want to limit the shape of focus areas to mere circles. In fact, some of the more advanced design- and interaction-concepts for AudioPhield require arbitrarily shaped magnification regions. Thus, the just developed circular fisheye lens serves as basis and starting point for the magnifying distortions in AudioPhield.

Figure 5.5 illustrates the current state of the design process.

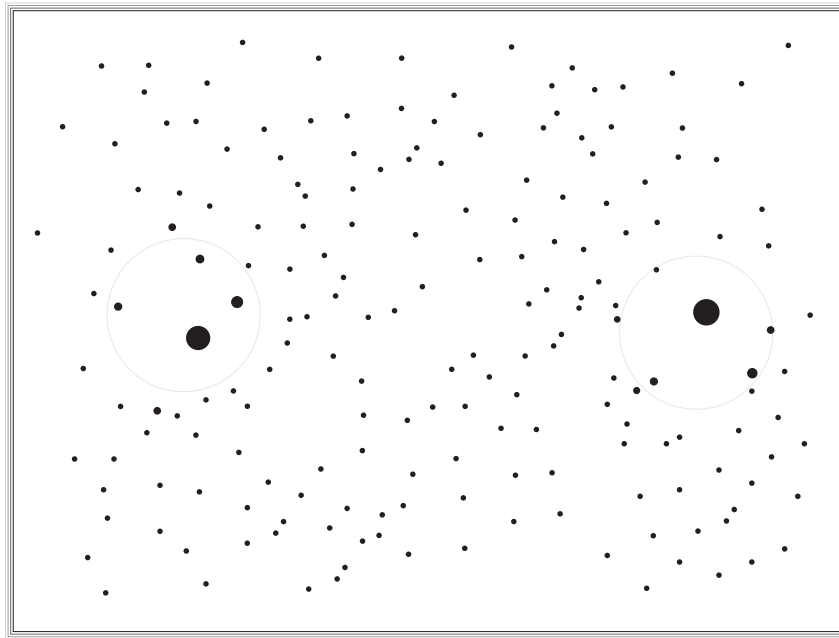


Figure 5.5: Illustration of an interface with fisheye views

Figure 5.5 also reveals another weakness of fisheye-views: It is rather hard to estimate how the distortion affects individual icons. As a consequence, users may have difficulties to judge the closeness of two songs, which is in the focus area one of the most important information encodings. Our approach to at least diminish this problem is to make the effects of the fisheyes as transparent as possible. Based on the recommendations in chapter 6 of Carpendale's dissertation [13] we have therefore devised the following measures:

1. **Color:** The background of the plane is colored according to the magnification strength. Therefore, the saturation of a color, which should not encode other information anywhere in the interface to avoid misconceptions, visualizes the distortion intensity. The visualization should use a fairly small maximal saturation to ensure that the contrast between icons or labels in the focus area and the background does not suffer.
2. **Grid:** The distortion of a regular pattern is easily readable. So, a grid, which is also affected by fisheye distortions, is to be displayed in the background. Again, the

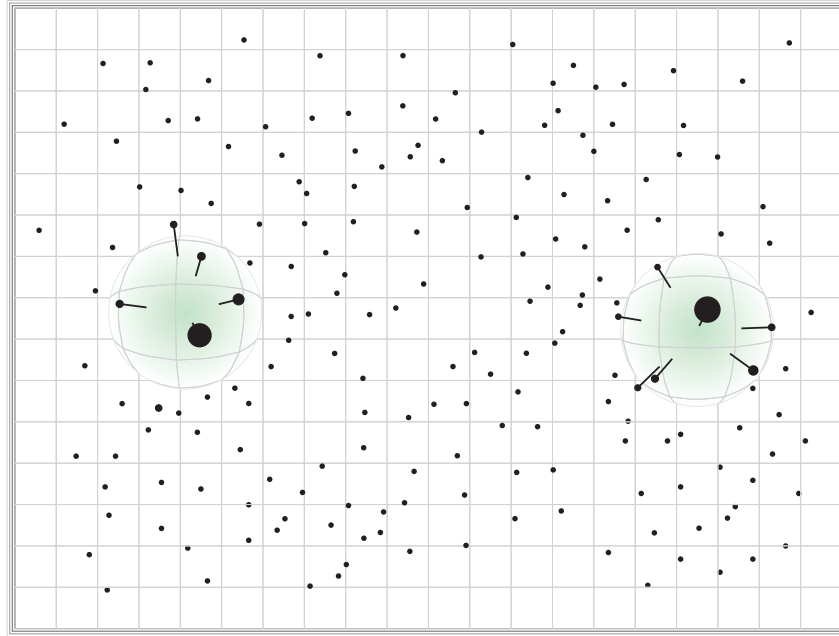


Figure 5.6: Illustration of an interface with visually emphasized fisheye views

danger of cluttering the interface arises. Thus, the grid should be painted subtle enough to not distract from the actual visualized information and distinctly enough to aid the user in comprehending the magnification.

3. **Origin-lines:** This technique shall especially improve the perceivability of the proximity of magnified icons. Lines are displayed that connect the current position of an icon in the distortion field with its original, undistorted position. Since lines are already used to connect songs, which belong to the same album, it is necessary to depict these lines differently. We use therefore color-coding as a simple solution.

With these measures the user will hopefully be able to understand which icons are affected and in which way. Figure 5.6 outlines how the interface should look like at this stage.

5.2.2 Information Encoded in the Icons

Now that the general layout of AudioPhield is defined, it is time to think about how the song icons should look and what information they should encode.

First of all, a way needs to be found to identify a song on the field without actually playing it. As seen in 3.1.2, the crucial information here is the name of the artist and the song's title. Since these attributes are necessary for browsing, they can not only be displayed on demand; having to somehow interact with every single icon just to find out which track it represents would make browsing quite cumbersome. Because it is on the other hand impossible to show artist- and title-information for each icon, we couple the display of these identifiers with the magnification inside the focus areas: They are only shown if the magnification of an icon exceeds a certain threshold. To augment the outer regions of the focus area, too, the technique is subdivided with a smaller threshold that only triggers the painting of the title. We chose the title instead of the artist for this step because we deem it more able to identify a song.

Usually, starfield-like views show simple labels containing identifying text above or beneath the corresponding item. This technique is impracticable in AudioPhield because

of a peculiarity of tabletop displays: Users can easily move to different positions around the table – so, there is no universal direction texts can be aligned to. Many tabletop systems provide the user with special interaction techniques to orient the contents of the interface to their needs (e.g., [33, 35, 70]). Others rotate the content instead (or in addition) according to its position on the table, assuming that every user interacts first and foremost with the area of the tabletop immediately in front of her (see, e.g., [37, 77]). For AudioPhield, we propose a different approach that makes the whole interface direction-less. Instead of displaying the text oriented in a specific way, the system draws it on a circular path around the icon. In this way, we eliminate the need to provide the user with interaction techniques to reorient the content – which would have added unnecessary complexity to the interface – and make the text from every perspective equally well readable. If an icon is central in a focus area, it is surrounded by two text-circles, the inner one for the artist, the outer showing the title. To improve readability, the radial strings rotate around the icon. So, letters belonging to different songs should be better distinguishable here than in the static case. See Figure 5.8 on page 36 for an illustration – compare especially 5.8(a) and 5.8(c).

This kind of display might also support the conversation between group members: Parts of the interface oriented to fit just one user are considered as “private”. Because of that, it is considered offensive to disturb another user’s private area – already looking into such a region might feel impolite²⁵. Emotions of this kind should barely surface with radial texts. So, it should seem acceptable for users to observe what other users are currently looking at – and to comment on this music.

We entertained for some time the idea to use album art as icon graphics. As mentioned in 3.1.2 these images could aid the identification of songs greatly. It would also be easier to find songs of the same album – it could in fact make the connecting lines, which currently show this togetherness, dispensable. On the other hand, the album art would arguably only be of help in strongly magnified areas. Worse, it prevents any form of color coding. Because this display dimension is highly valuable to present other data, the idea was finally dismissed.

Information to identify music is now incorporated into the interface. However, there are other important attributes to a song that should be visualized:

Ownership: One of the most interesting features of AudioPhield is that it is able to show private music libraries from different people at the same time in the same visualization. However, at this stage in the design process there is no cue present to let the users know to whom a song belongs. We chose to use colors to encode the ownership because from the remaining possible display dimensions (color, shape, size, motion, and texture) color is the only possibility that should still be preattentive processable when the icons are not in a focus area and thus quite small. So, users are empowered to compare their collections – and thus their musical identity (see 3.2.1) – almost instantly just by estimating where on the field icons of all users are equally distributed and where the icon densities differ (see Figure 5.7). Therefore, the colors should be as distinguishable as possible. So, we use colors with very different hues. It is to be expected that songs are contained in more than one library. For these, the icons are divided like a pie-chart into equal parts so that there is one – appropriately colored – part for each owner (see Figure 5.8(a)). Radial texts floating around the icon should not be partitioned in the same way because their readability would be damaged. Instead, the system displays the text with the average color of all owners. The split-approach can only work when the song icon is magnified; otherwise, the icons are

²⁵The influence of territoriality and orientation in collaboration around a table is discussed profoundly by Kruger *et al.* in [49].

too small and the partition is imperceptible. As solution in this case, non-magnified icons are colored arbitrarily in the color of one of the owners. Then, for each additional owner a ring enclosing the icon (and previous rings) is added. In this way, it should be clearly visible that a song belongs to more than one user – it is, after all, now encoded as color *and* size (see Figure 5.8(b)).

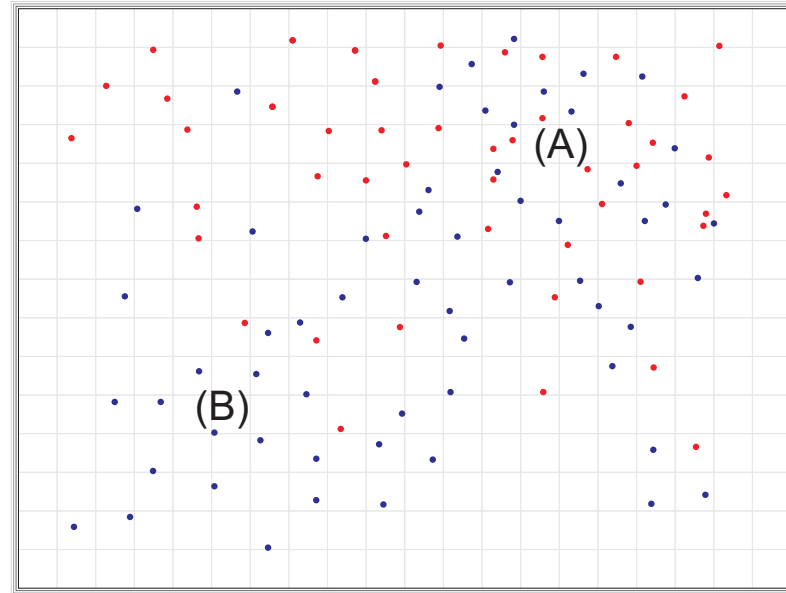


Figure 5.7: Illustration of an interface containing two music libraries.

Note, how areas, where the libraries are very similar (A), are clearly distinguishable from areas where one user has more songs than the other (B).

Recency of last playback: Section 3.1.1 and 3.1.2 revealed that users usually maintain an active set of music, i.e. they play some songs very often compared to the remainder of the library. Thus, AudioPhield should also visualize how recently a song was played. We chose for this purpose the remaining channel of the display dimension color, saturation, because the other still available display dimensions are inapt here: Visualizing the recency of the last playback in form of shapes would limit the presentable values strongly (see Table 5.1) and seem counterintuitive. Furthermore, users would only be able to discriminate the shapes when they are in the focus area. Thus, playing her working set would force a user to scan the entire visualization looking for the right shape. Encoding playback recency by modifying the icons size would not have these flaws. However, size-encoding is here not advised because it is already used to signal the number of a song's owners, and because it must be limited to the non-magnified areas: The smoothly ascending magnification in the focus scope of the fisheye distortion makes it virtually impossible to compare the size of different icons. This problem is not present at the visualization of ownership above because the size encoding is limited to unfocused areas. Thus, utilizing saturation is the only way left to visualize this important data. Unfortunately, not the full range of possible saturation can be utilized here because the lower the saturation the closer the color is to gray. So, too low values make it hard to judge the color of an icon and thus the owner of the represented song. If a song belongs to more than one user, the saturation coding applies to the different parts separately; for the radial texts, the highest saturation is taken to ensure good readability. Actually, AudioPhield does not use saturation here but opacity. The effect is about the same but it is a little more pleasing to the eye. See Figure 5.8(c) for an illustration.

Rating: Most popular programs for music playback offer the possibility to rate a song, usually on a scale from one to five. This data is obviously also valuable for the casual browsing in AudioPhield. To visualize it, we chose a display dimension that should not interfere with the amount of already integrated visualizations: Rotation. The icons, as well as the radial texts around them (if present), spin around their center according to the rating of the songs they represent: Icons of low rated tracks rotate very slowly while highly ranked songs cause the icon to spin at a higher pace – but not so fast that it makes the radial texts unreadable. Only clockwise rotation is used: Albeit mapping the ranking on a scale ranging from fast counterclockwise movement for low ranked songs to fast clockwise spinning for high ranked ones would enhance the discernibility of adjacent ranks; the speed of the rotation, which is perceived faster and more easily than the rotational direction, would visually couple songs of different ends of the ranking scale closer than, e.g., a song ranked very high and one ranked medium – misconceptions would be inevitable. In the case of multiple ownership, the average rating is used to determine the rotation speed (see Figure 5.8(d)). This should support the users further in comparing their taste in music: Fast-spinning dual-colored icons are a clear indicator for overlapping preferences. Unfortunately, this visualization technique is rather subtle due to the small bandwidth of usable paces. Also, it is nearly impossible to perceive the rotation of the tiny unmagnified icons. Thus, the visualization is essentially limited to focus areas. Regardless of this drawbacks, the spinning interferes barely with other visualizations and adds, because of its subtlety, virtually no complexity to the interface. So, although it transports little information, the visualization is kept as part of the interface due to its low cost.

The encoding of playback recency and rating in the presented way seems reasonable to us and leans on observations of typical access-strategies for private music libraries (see 3.1.2) – but it is eventually arbitrary. It is entirely possible that a song’s rating is more important to collaborative casual browsing than its playback history and should therefore be visualized through the more articulate channel opacity. Also, there is another variable not taken account of: The overall play count, i.e. how often a song was played since it was added to the library. It is at this time not visualized because we assume that this value correlates strongly with the rating – in fact, it might be a good idea to compute a rating estimation automatically from the play count if a song was not rated before. We could also use shape or texture as one of the few remaining display dimensions to additionally encode this information. This would, however, interfere with the already used dimensions and make the interface, which already seems rich enough of encodings, more complex. Thus, all icons are drawn with the same shape, a disk featuring the familiar “play”-symbol, because this shape fits best to the radial texts and once more pronounces that AudioPhield’s interface is supposed to be direction-less. However, only user tests can confirm that the above presented design decisions are sensible. Figure 5.8 presents an illustration of the different encodings introduced in this section.

5.3 Enable Simultaneous Interaction on a Tabletop Display

The previous discussions defined exactly what kind of information is to be visualized in which way and how the interface on the whole should look. However without means to interact with this imagery, all of the techniques above are rather useless. This section will introduce the different interaction concepts of AudioPhield. These concepts are designed to handle some peculiarities of the system’s setup as good as possible:

1. **Direct-touch:** The tabletop display, unlike common interaction devices as mice, allows users to directly touch any entity they wish to interact with. This enables the

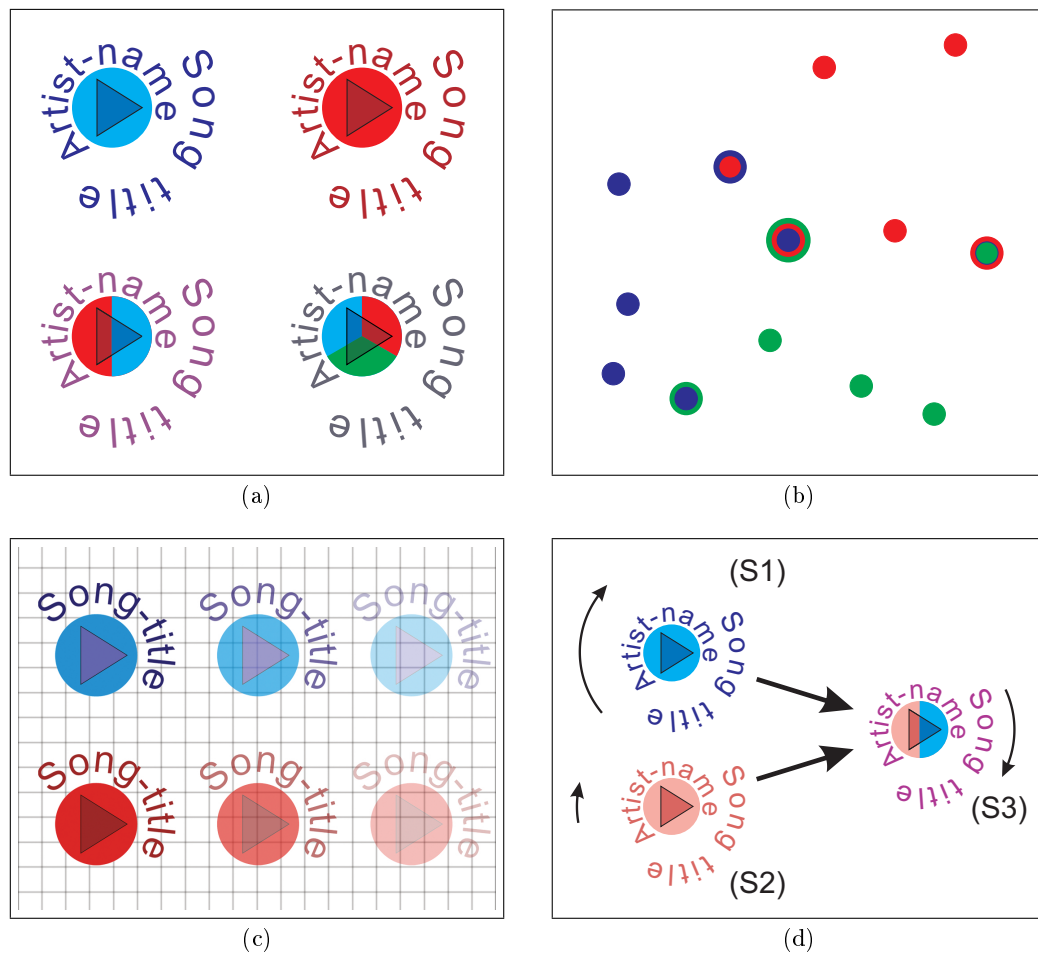


Figure 5.8: Illustration of the different encodings incorporated in the icon display.

(a) Color codings to indicate ownership of one or multiple users in maximal magnified icons.

(b) Color codings to indicate ownership of one or multiple users in unmagnified icons.

(c) Color codings to indicate a song's recency of playback from very (left) to not recent (right). The icons are only moderately magnified, so only the title texts are shown.

(d) Encoding of song rankings through rotation speed (indicated by the radial arrows). (S1) and (S2) are visualizations of the same song in different libraries. They differ in recency of playback and rating. (S3) shows the representing icon for the song when both libraries are combined on one interface.

creation of interfaces that feel more intuitive and “natural” than common desktop GUIs²⁶. However, reaching out and touching a spot on the table means that the manipulated control is occluded by the finger and other interface parts by the arm. Also, according to Buxton’s “three-state model of graphical input” in [10], direct-touch screens are just “two-state interfaces”. This means essentially that the usual distinction between a tracking-interaction (just moving the cursor) and a dragging-interaction are impossible – there is no mouse clicking to determine the mode. Finally, the interaction precision is rather low in comparison to traditional interaction devices (see chapter 6.1).

2. **Multi-touch:** The hardware, on which AudioPhield shall operate, does not limit interactions to a single point; in fact an arbitrary number of simultaneous interaction spots is supported. To exploit the possibilities offered by this, new ways and metaphors of interaction are necessary.
3. **Table-top:** The interaction-area is integrated in the surface of a table of noticeable size. Thus, it is to be considered that a user may not be able to reach every part of the interface from her position. While in our case the users are standing around the table and can so easily change their position, the operation of the system would feel cumbersome if they were forced to walk around the table to execute frequent interactions. Finally, while people can support themselves with one hand on the rim of the table to reach further out for single-handed interaction, this option is unavailable for two-handed input. So, the maximum reach in the latter case is significantly shorter.
4. **Multi-user:** AudioPhield is designed to enable social browsing of music libraries, i.e., it needs to cope with interactions that stem from different users and are not linked to achieve the same goal as it would be the case in a single-user environment. As a consequence, the interaction techniques need to be modeless²⁷ because any change of how current inputs are to interpret will most likely disrupt another user’s activities. However, local modes, i.e., mode changes limited to small and clearly identifiable parts of the surface, are possible.

5.3.1 Manipulations of Focus Areas

As section 5.2.1 presented, AudioPhield shall feature fisheye views to create focus areas at arbitrary positions on the interface. Since *browsing* music libraries is the fundamental purpose of the system, it is important that moving a fisheye area is as easy as possible.

We developed a number of different concepts to to create, move and destroy focus areas. Of these, we will present three in the following.

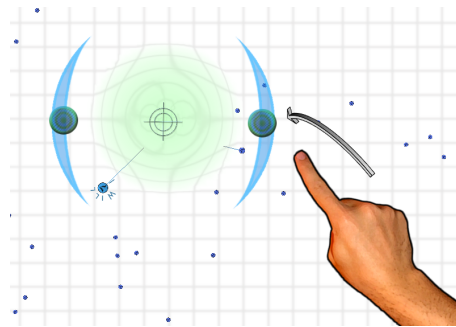
ZoomFrames

This approach uses the button-metaphor known from common desktop GUIs. Because of this close relation to usual mouse-controlled interfaces, it should feel familiar to novel users. The focus areas are here enclosed in a circular frame that has two handle-spots attached to it at opposing sides. All interaction with the ZoomFrame is executed via these handles: To move the frame, the user needs to press a handle and drag it to a new spot (see Figure 5.9).

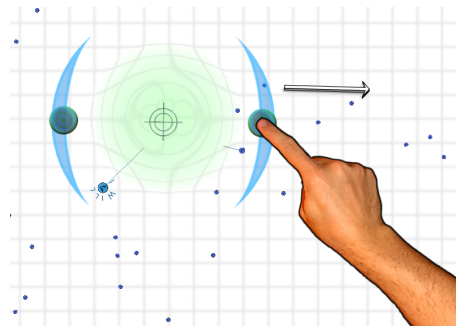
Operating both handles at the same time allows controlling the size of the frame and hence the fisheye. The strength of the distortion and the size of the frame are thereby coupled. This coupling is essentially included because of interaction experiments which

²⁶‘GUI’ is the abbreviation of ‘Graphical User Interface’

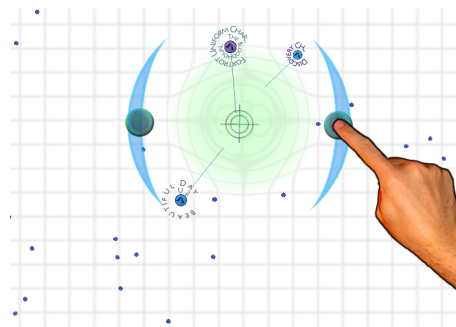
²⁷See [22] for a complete introduction to “mode” in interface design.



(a)



(b)



(c)

Figure 5.9: Movement of a ZoomFrame.

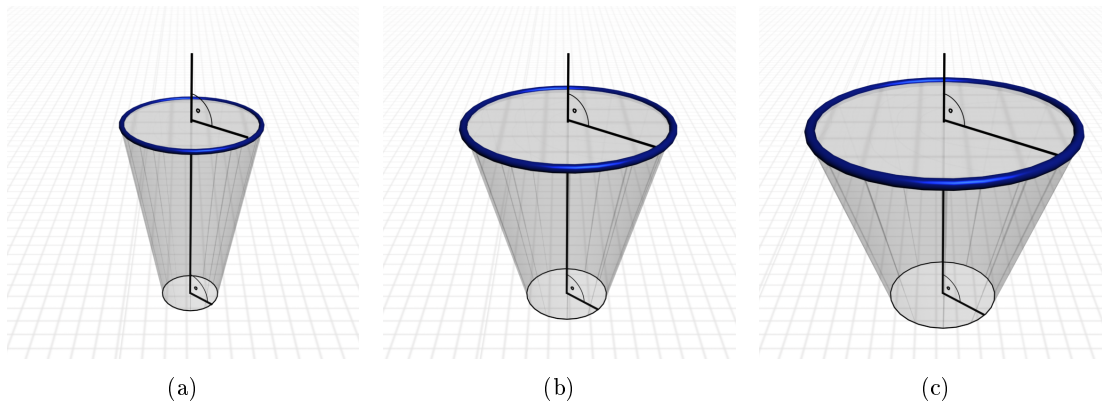


Figure 5.10: Illustration of the coupling of frame size and zoom in ZoomFrames.

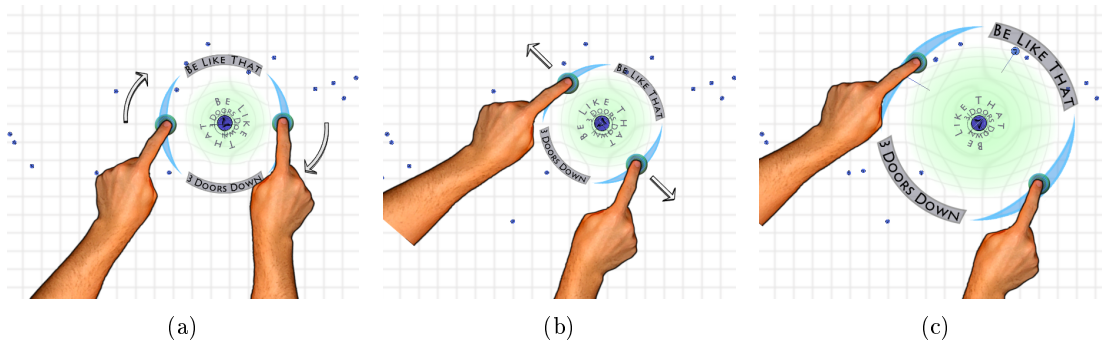


Figure 5.11: Rotation and resizing of a ZoomFrame.

showed that users expect a larger lens to have a stronger magnification. The possibility to adjust to some degree how wide the magnification spreads out a cluster of song icons should also be valuable to enhance the legibility in areas of high icon density. To make larger ZoomFrames also offer more overview, the size-magnification-coupling is loose enough that the area magnified by the frame also depends on the frame size (see 5.10 for an illustration).

Touching and moving both handle-spots also allows to rotate the ZoomFrame. As with a real lens, this has no effect on the distortion and magnification. However, to not waste this natural interaction technique, we utilized the fact that one frame belongs usually to only one user, and added two labels to the ZoomFrames that present artist- and title-information to the song currently in the center – indicated by hairlines – of the frame. The user can make the text on these labels optimal readable for her by modifying the orientation of the frame. This technique should be quite valuable in areas with a high icon density. Figure 5.11 outlines the presented size- and rotate-interactions.

To enable simultaneous browsing, the interface must not be limited to a single ZoomFrame. Therefore, users can create a new frame at any time by pulling it from one of the creation fields, which are located at opposite corners of the interface. Since display space is a scarce resource, and since this interaction should occur rather infrequently, two such creation fields should suffice. Consequently, a function to destroy ZoomFrames needs to be integrated. For that, the user just has to minimize the frame: If its size falls below a certain threshold, the frame is painted red. This appearance-change is necessary because the minimizing triggered a new local mode: As soon as both handles are released, the frame vanishes – while usually this input has no consequences. See Figure 5.12 for an illustration.

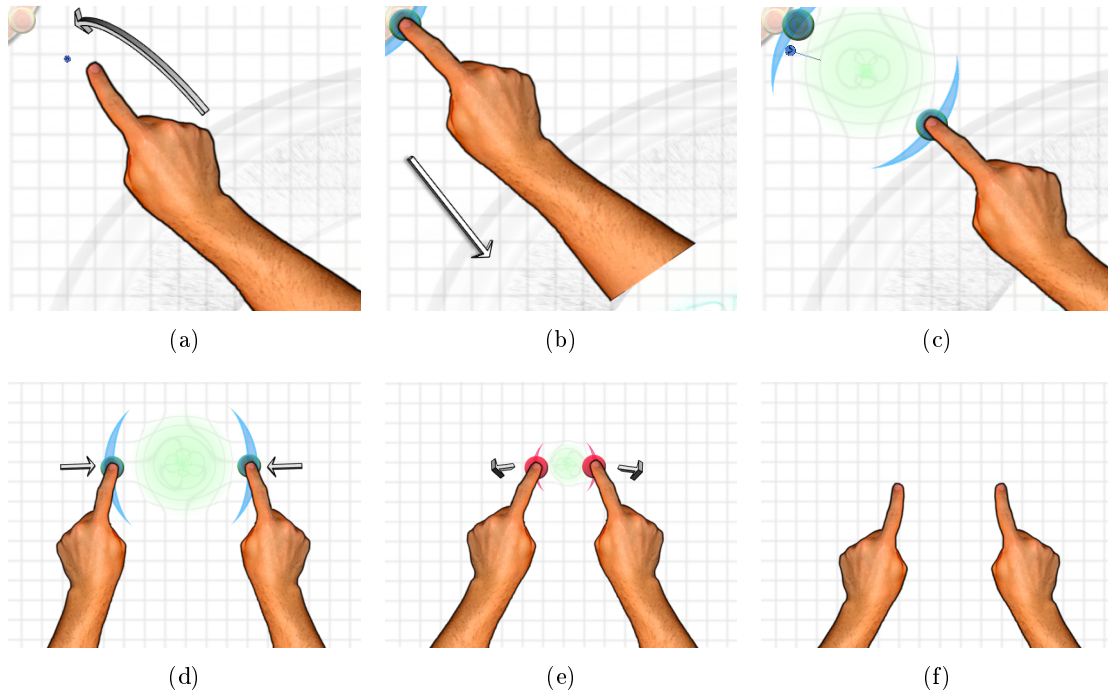


Figure 5.12: Interactions to create ((a)–(c)) and delete ((d)–(f)) ZoomFrames.

SoapSpots

For this interaction technique initially the metaphor of a water surface with small bubbles (the song icons) swimming on it was used. Where a soapy finger touches the surface, the surface tension decreases and objects swimming in the water drift apart. The interface was to act analog: Users could influence the distortion and magnification anywhere by just touching the tabletop display at arbitrary places – the longer they let their finger linger at the same spot the stronger and further reaching the magnification becomes. Then, when an area receives no more input, it returns slowly to its original, undistorted state.

This initial design revealed some major flaws: The time a distorted area needs to return to its normal state can just not be set right. Either, when the attenuation proceeds rather slow, recently browsed areas are so distorted that it is virtually impossible to access them; or, if it proceeds fast, the user needs constantly to touch the spot which she is currently observing anew. Also, occlusion is a problem: Focus areas are obviously the parts of the field that a user is currently interested in; unfortunately, to create the focus she has to touch – and therefore occlude with her hand – exactly this region.

So, the concept was redesigned into the following: Users might still create regions of focus – here called “SoapSpots” – just by touching an arbitrary point on the interface. In contrast to the original approach, the summoned fisheye lenses are now mobile. When a user touches the interface anywhere inside such a lens, it sticks to her finger and can be moved anywhere. This solves, on the one hand, the occlusion-problem since a focus area needs not to be touched centrally, and, on the other hand, the problem of inaccessibly distorted display parts – after all, the lenses can simply be moved away. Figure 5.13 demonstrates the creation and movement interactions.

Unfortunately, the simple means to adjust the size of a SoapSpot ceased to exist with this modification, too. As a replacement, the user can modify a lens’ size similar to the procedure used for ZoomFrames: When there are two interaction points inside the area of a fisheye, distance changes between these points are interpreted as scaling operations.

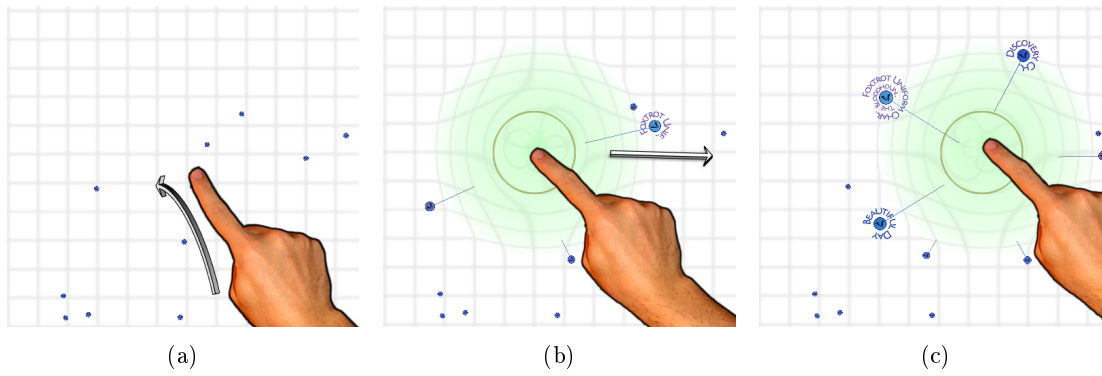


Figure 5.13: Creation and movement of a SoapSpot.

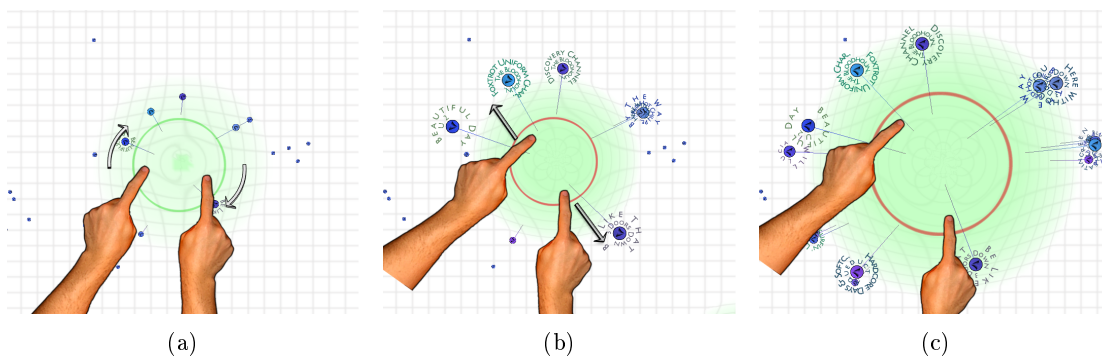


Figure 5.14: Rotation and resizing of a SoapSpot.

Unlike ZoomFrames, SoapSpots accept input anywhere inside their contour and not just at dedicated handle-spots. Again, the size of a fisheye lens and its magnification are coupled (see Figure 5.10), for the same reasons as above. With SoapSpots there is additionally another way to modify this magnification: Rotary two-pointed interactions directly influence this factor. Thus, a SoapSpot can be set to minimal distortion for easy navigation or high magnification to separate very close icons better. An demonstration of the just discussed input types can be found in Figure 5.14; although the illustration shows two-handed input, there is no need to do so – one-handed interaction with two fingers should work just as well.

Tests during the implementation revealed quickly that SoapSpots require an additional visualization of their interactive area: Users often intended to move SoapSpots but touched outside of the input-accepting area of the spot and so created new SoapSpots instead – much to their frustration. The visualization of the fisheye lens is informative of how strong certain song-icons are affected by the distortion but due to its diffuse borders does not suffice in hinting the margins of SoapSpots. Thus, the active area is now clearly surrounded by a circle, which is additionally also color-coded to show the current zoom-level of the focus area. All illustrations above regarding SoapSpots already contain this ring. See especially Figure 5.14 for the ring's color-coding.

Illustration 5.15 demonstrates the way of interaction to remove SoapSpots from the interface: Redundant spots can be combined with others. Therefore, just one SoapSpot needs to be brought close enough to another (regardless if that spot is currently interacted with) so that their boundary rings overlap to a certain degree – then they are combined to one. If only one of the two spots was touched before the combination then the moving SoapSpot assimilates the other one and the latter effectively just vanishes. Were both

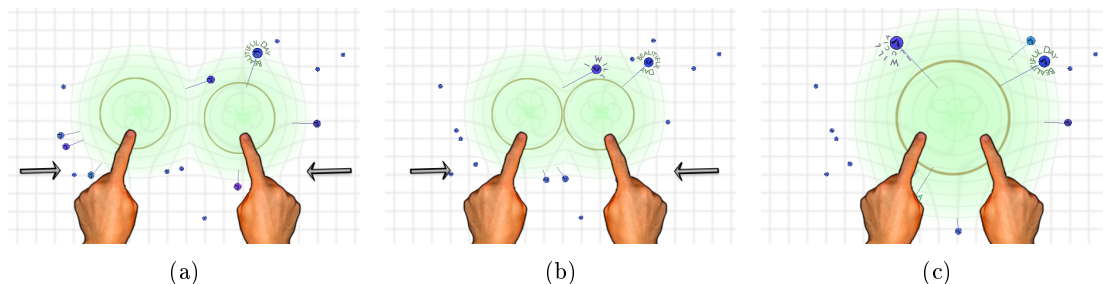


Figure 5.15: Combination of two SoapSpots.

touched, the combined SoapSpot’s size is determined by the distance between the two touch-points and its zoom-level by the average zoom of the source spots. This interaction technique is especially useful, if the user creates accidentally a new SoapSpot instead of rescaling an existing one: She just has to bring her hands (or fingers, respectively) quickly together before expanding them again. So, with little delay and without finger lifting the intended interaction is executed, and the original mistake is corrected. However, also unintentional combinations of SoapSpots handled by different users may occur. User tests will have to show if emerging social protocols suffice to avoid this problem, or if ways to prevent it need to be found.

Although this method to handle focus areas is still called “SoapSpots”, it incorporates hardly any aspects of the original metaphor. Without this consistent and familiar linking element, its learning curve might arguably be steeper in comparison to the initial concept and the ZoomFrames. However, with the directly manipulable zoom-level, it also offers more control over the fisheye lenses.

CookieDough

The interface colorfully named “CookieDough” is our most uncommon and innovative approach to manipulate distortions and magnifications in AudioPhield. It has its name from the underlying metaphor of a baking tray covered with a ductile dough flecked with spots of chocolate dough. If dough of this kind is kneaded, the chocolate spots are distorted and stretched. In the interface the whole screen corresponds to the cookie dough and the song icons are the chocolate spots. Interacting with the virtual dough should thereby resemble touching real dough as close as possible. So, when a user touches the surface at any spot, the dough “sticks” to the interaction points. Now, any movement stretches and condenses the virtual dough and so the song icons, too – distorted, magnified areas emerge. Is the input stopped, the virtual dough relaxes and the regions of focus slowly vanish.

Unlike the previous approaches, the user is thereby not limited to using just the tips of her fingers but may use the side of her hand (as illustrated in Figure 5.16) or even her whole arm. Also, her interaction is not interpreted as punctual but the actual shape of the touched regions are analyzed and used as a feature of the input.

With this immediacy and the consistent metaphor, the CookieDough interface should feel more natural and accessible than any of the above presented interfaces. However, there are drawbacks in this concept, too. Like in the original SoapSpots-interface the problem of the relaxation speed arises: Too fast, and users are forced to redo their last magnification input repeatedly, too slow, and areas stay inaccessible for too long. Obviously, the metaphor – or rather the used tabletop technology – reaches here its limit: With real dough, one could smoothen it by flattening it softly. The virtual system can not distinguish between strong and soft input, and can therefore not adopt this way of interaction.

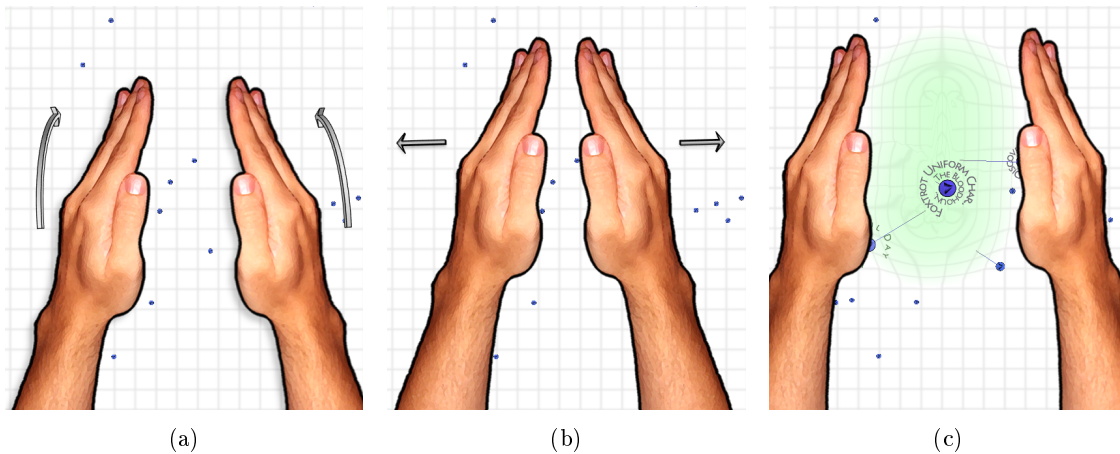


Figure 5.16: Illustration of the CookieDough interface.

Another problem, however, caused us to not implement it for now: The central purpose of AudioPhield is to support users in browsing music; the CookieDough interface on the other hand, albeit apparently very useful to create arbitrary focus areas in a very plausible way, seems hardly appropriate for this use case because it offers no way to just move a focus area – instead a new one must be created by two-handed input that is considerably more laborious than drawing a single finger over the surface. Worse, the direct neighborhood of a stretched zone is logically condensed and therefore harder to stretch – the user is forced to obstruct her own browsing.

These for now unsolved problems and the fact that the CookieDough interface would significantly complicate the implementation process due to the fact that we would have to change the used computer-vision library (see section 6) extensively, caused us to exclude it from the current prototype of AudioPhield presented in this thesis.

5.3.2 Music Playback Control

The above interaction techniques grants access to much information about a private music library, including how a piece of music might roughly sound. But how the song *actually* sounds remains a secret so far. This is to be changed in this section.

Simple Playback

Playing a song is the central part of any software that is supposed to handle music and should therefore be made as easy as possible. In a desktop environment this would mean that a song is played by simply clicking it once. In the case of direct-touch interfaces, where no explicit click exists (see the list of peculiarities in the beginning of section 5.3), the closest equivalent to a single click would be a single tap, i.e. a touch interaction with minimal movement that lasts only for a short time ($< 150\text{ms}$). This was also our first approach: A single tap on an icon triggered the playback of a song. However, continuous testing during the implementation showed that a tap differs too little from other short interactions. Thus, unintended playbacks happened frequently and led to much frustration – understandable: After all, the mistakenly played song did usually not match the user’s current mood and was therefore the “wrong music” (see section 3.1.2). In addition, issues of the tracking of user input worsened the problem: Occasionally the tracking is lost, i.e., one dragging movement over the interface is interpreted as two consequent inputs (see chapter 6.2.1, page 51). So, if two losses happen in close succession, a phantom input occurs that looks to the system exactly like a tap.

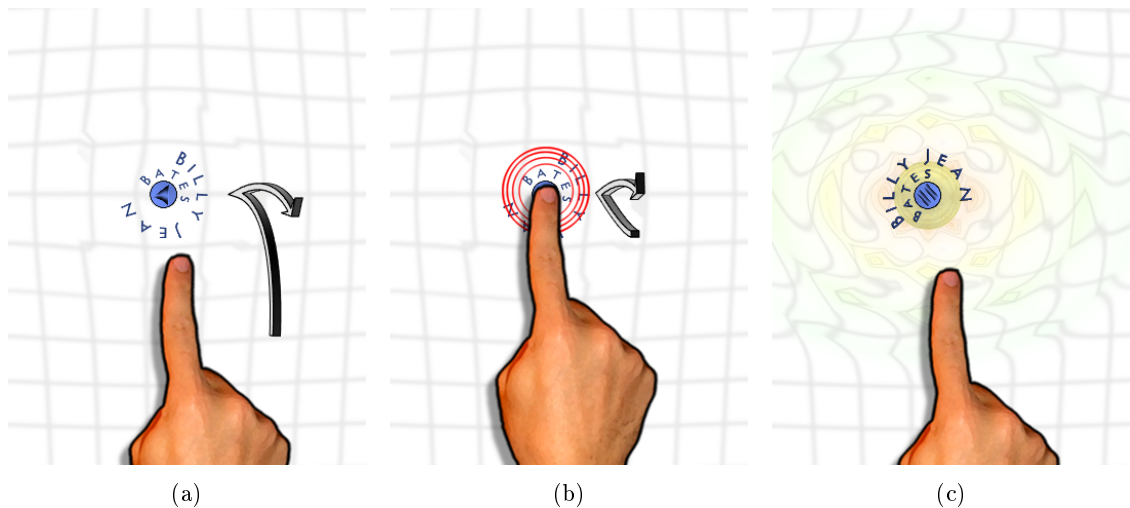


Figure 5.17: Illustration of the double-tap interaction to start playback of a song.

Thus, while the playback of music should be easy to trigger, it is also important that it is hard to trigger unintentionally. Our idea to meet this requirement is to use double-taps, i.e., two short taps, as defined above, with a short break of less than 500ms between them. The maximal timespan between the taps was chosen arbitrarily but seems reasonable: It should be short enough to not occur incidentally, and long enough to be easily executable. Tracking losses could still mimic this interaction, but this is very unlikely because *three* tracking losses in close succession are rather rare. But when they appear, then usually during faster movements – and two short taps in different spots are not confused with a double-tap.

Since there is no clear tactile feedback if a tap was recognized by the system, at least clear visual feedback seems recommended. AudioPhield displays therefore a pattern of concentric circles around a tap. These circles stay visible but shrink for as long as a second tap would be interpreted as the second part of a double-tap. A successful double-tap is not visualized in any way because its effect is clearly visible: If a song is currently played, the associated song-icon is painted with a yellow disk around it to mark its special function. This disk might not suffice if the user moves her focus area away. Thus, the icon of the playing song becomes also the center of a subtle visualization of the playing music in the background, and its magnification can not drop below a certain threshold. These measures should ensure that the users are never in doubt which song is currently playing and where it is to be found on the interface. As another change, the symbol on the icon changes from “play” to “pause”. This indicates already what a second double-tap on a playing song provokes: The song is paused and the icon switches back to the “play”-symbol because another double-tap resumes the playback. See Figure 5.17 for an illustration of the playback-triggering interaction.

Scan-Playback

Sometimes, a user will not recognize a song solely by its artist- and title-information, e.g., when a song is new or does not belong to her library but to the collection of another user, which is visualized on the same interface. Also, she might just want to orientate herself on the field and just listen to *any* song in an area to learn which kind of music is to be found there. In these cases, users are only interested in listening briefly into a song instead of a full playback. To meet this interaction need, AudioPhield features a special scan-playback function. By tap-holding, i.e., touching a position for a specific minimum

time²⁸ without moving the touch-point, a song icon the scan-playback is activated. Now, the song is played, beginning at a third of its duration to skip intros, as long as the finger is not released. If it is released, the playback stops instantly. After the activation time of the tap-hold interaction there is no need for the touch-point to stay in the same place since it was already identified correctly. Thus, it may be moved freely – and needs to be for the winding interaction described below.

So, there are now two ways to play music for different purposes in AudioPhield: One serves to play a song the users actually want to listen to, while the other should only provide a glimpse of a song for browsing- and navigation-issues. Both kinds will occur during a usual interaction session – and in most cases at the same time, as it is to be expected that some music plays all along, maybe in the center of the users’ attention, maybe just as background noise (see page 16f). Thus, scan-playback should interfere with the full playback as little as possible. In the current design, this is achieved by pausing the primary playback during a scan-interaction and resuming it instantly afterwards. We implemented also other possibilities, in which both songs are playing at the same time: E.g., the fully playing song is just muted, or the songs sound from different speakers. However, in preliminary tests, the results of these approaches seemed rather annoying because the songs could not be clearly distinguished. On the other hand, it was appreciated that these solutions sound less “skippy” than the pause/resume-approach. Therefore, more profound testing is necessary.

Also, it might happen that two users want to use the scan-playback function at the same time. We found two solutions to solve the conflict: Firstly, the earlier playback is interrupted and terminated by a new one; secondly, new playback attempts are delayed until earlier scans are finished. The first alternative is used as default in AudioPhield because it is consistent with the full playback and it does not disappoint a user’s expectations by a mode change: A tap-hold interaction triggers scan-playback, regardless of other circumstances. In any case, we expect conflicts of this kind to appear rarely because of emerging social protocols to avoid them in the first place.

The visualization of the tap-hold interaction is similar to the visual feedback of a double-tap: It is also indicated by a pattern of concentric circles. In contrast to the latter, the pattern here is differently colored, and it is growing until the interaction is complete or aborted by releasing the touch-point or moving it. Both differences should ensure that the tap-hold interaction is understood as similar to the double-tap but also prevent mix-ups. The visualization of a scan-played song equals the one for fully played songs with the difference that now the disk under the song is gray instead of yellow. Also, the playback starts here already in the “seeking-mode” (see below). While this is technically not part of the scan-playback-visualization, it serves as a further visible indicator. Figure 5.18 illustrates the discussed tap-hold interaction.

The same figure also demonstrates a case of “details-on-demand”, as postulated by Shneiderman’s principle (see paragraph 5.2.1): The lines that connect songs of the same album appear only when a song is fully- or scan-played (see panel (c) in the figure). Thus, we interpret – hopefully reasonably – the playback of a song also as a wish for additional information that triggers the “on-demand” functions. At this time, these functions are limited to the aforementioned lines – in the future, however, further functions might be added if user tests reveal according needs.

In-song Seeking

To keep the user interface as simple as possible, there was initially no function to jump

²⁸Tests during the implementation indicated 750ms as a good value. Shorter time spans caused frequent misconceptions, while longer spans slowed the interaction unnecessarily.

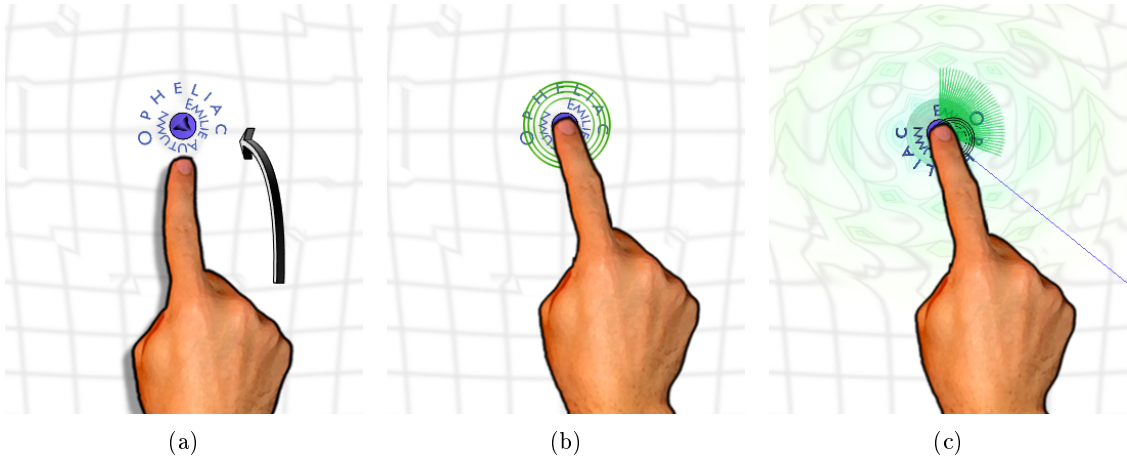


Figure 5.18: Illustration of the tap-hold interaction to start scan-playback.

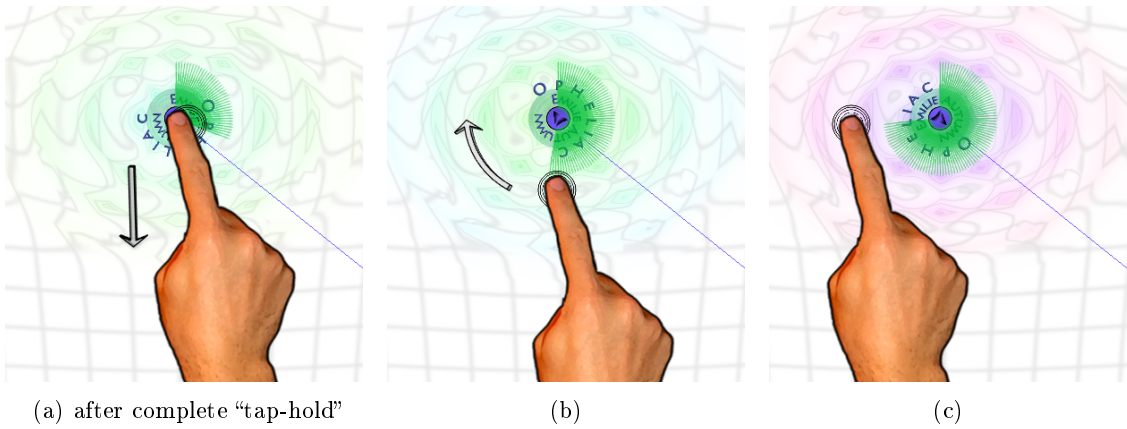


Figure 5.19: Illustration of an in-song seeking interaction.

to a specific position in a song included in the interface. The first preliminary tests, however, demonstrated clearly that it is urgently needed. The typical cases of need for in-song seeking were to get a better overview of an unknown song, and to present the best, worst, unusual, or otherwise special part of a song to another member of the group. Since these purposes resemble the information need the scan-playback was developed for, the in-song seeking function is triggered in the same way as the latter: As soon as the tap-hold interaction on a song-icon – regardless if the icon presents a track already playing previously or just started in scan-mode – is complete, the seeking function is activated. A green disk sector, which visualizes the current playback position, appears. Now, when the touch-point of the tap-hold interaction is moved into this disk, the playback jumps to the indicated place. The full disk represents thereby always the duration of the complete song, i.e., the timespan represented by one angular unit varies with different song lengths. To enable input of high precision, the user can move the controlling touch-point as far from the song icon as she wants: Logically, with increasing distance, the angular change caused by the same spatial translation decreases. Figure 5.19 illustrates the necessary interaction to jump in a song first to the middle and then to about three quarters of the playback duration.

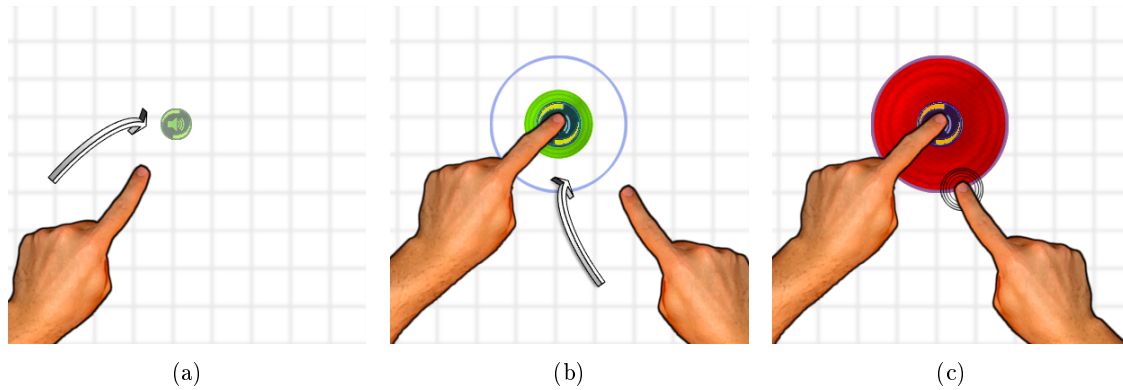


Figure 5.20: Illustration of an interaction to raise the volume.

Volume Control

The fact that not all songs are recorded at the same level, and that some songs just need to be played *louder* to make them sound right, demonstrates that AudioPhield must feature a possibility to adjust the playback volume. We developed therefore a “VolumeWidget”, a control solely dedicated to visualize and modify the current playback volume. To operate it, the user has to activate the – again circular designed – widget by touching it anywhere. Then, the widget switches to its active mode and is painted notably larger than before for as long as the input lasts. It is now also surrounded by circle that indicates maximum volume. The space between the outer rim of the widget and this circle is used as input area to specify the desired volume: The further this area is touched by a different finger from the center, the higher the volume is set. A disk, which grows from the center to the touch-point, serves as visual feedback for the current position. As an additional cue, the volume is also visualized by the color-coding of the disk, which uses a scale from green over yellow to red. Figure 5.20 demonstrates the interaction necessary to change the playback volume. In the illustration, two hands are used for interaction; however, there is no reason why the same input should not also be executed with just one hand, e.g. with thumb and index finger.

This two-step process to adjust the volume may seem more complex than it needs to be, but there are good reasons for this design. Because no display space was to be wasted for an extra area to control the volume, the VolumeWidget is located as a movable entity above all other visualizations. It needs to be movable because if it is fixed anywhere, it might not be reachable for all users from their position and they would be forced to walk around the table every time they want to adjust the volume. Also, it might occlude some song icons. To move it, a user has just to touch and drag it to any position. The movement is the simpler interaction because of its error tolerance: If the widget is moved unintentionally, there is no harm done. If the volume is unexpectedly changed to its maximum by mistake, it might at least startle the users – and if it happens more often, annoy them. The two-step process above is far less likely to be executed incidentally than just touching the widget.

Although AudioPhield is supposed to be a multi-user interface there is yet only one VolumeWidget. This is due to the fact that there is also only one playback volume that affects all people in the room equally. To have more than one volume control would therefore just lead to confusion and occupy valuable display space.

As an alternative, we implemented a widget that interpreted angle-changes between two touch points on it as amounts of which the volume is to be changed. This design was inspired by volume knob found on many stereos and thus should feel familiar. Some

drawbacks, however, make it seem inferior to the above introduced concept: To avoid undesired changes (like sudden jumps to maximum volume), the rotation-angle necessary to modify the volume by a given amount should be chosen quite large – which may cause users to knot their arms or distribute their wished change to multiple interactions. As a solution, the widget could be modified in such a way that the touch-points no longer need to stay on the widget but may reach farther out. Then, however, a single tracking loss – which is more likely since two touch points are involved – terminates the gesture. So the user has to bring her fingers or arms together again to re-initiate the volume-changing interaction.

All the above mentioned interaction techniques are limited to browsing the music libraries and playing single songs. Many functions found in common music players are not included. For example, it is not possible to modify the rating of a song. This possibility was omitted because it is impossible to identify users; so, users might alter the ratings of songs that are not part of their collection – possibly against the will of the rightful owner. Also, there is no way to create or maintain playlists. In an environment, where users are interacting with music libraries and playing songs all the time, there just seems to be no need for playlists – it would probably be changed before the first song in it was fully played. In a single-user scenario, however, the used visualization could be of great value: A playlist could be created by simply drawing a line on the field²⁹; or, one could draw a circle around an area of songs to be played in random order. Future versions of AudioPhield might include functions like that.

²⁹ An exemplary implementation of such a function can be found in [39].

6 Implementation

After the previous chapters decided *what* AudioPhield is to be and *why*, this chapter examines *how* we implemented the result of this design process. The section is thereby structured as follows: Firstly, the technology and concrete setup of the used tabletop hardware is presented. Then, we discuss general issues of the software system as the used libraries and the overall architecture. This is followed by a section about selected implementation aspects, which address interesting or unusual solutions to some of the core-difficulties implicated by the design.

6.1 Hardware Setup

The touch-sensitivity of the used tabletop display is enabled by a technology called “frustrated total internal reflection” (short: FTIR), which was used for fingerprint scanners before Han presented its value for creating touch-screens in [32]. FTIR utilizes the effect that the reflection- and refraction-properties of a material are determined not only by its own refraction index but also by the refraction index of the surrounding medium. An array of LEDs emit infrared light of a preferably narrow bandwidth of wavelengths into the side of an acrylic pane. This light meets the interior surfaces of the pane in such an angle that it is totally reflected. Thus, the pane looks black to a infrared camera mounted below. If now the acrylic glass is touched anywhere, i.e., if the outer medium is replaced by one with a higher refraction index, the reflection in this spot changes from total to diffuse, and the light is scattered in all directions – including through the opposite inner surface of the pane. So, the spots where the glass is touched light up and are clearly visible to the infrared camera (see Figure 6.3(a)). In this way, the FTIR technology delivers the basis for computer-vision algorithms to find and track interaction points with high-contrast images of where a pane was touched. See Figure 6.1(a) for an illustration of the principle.

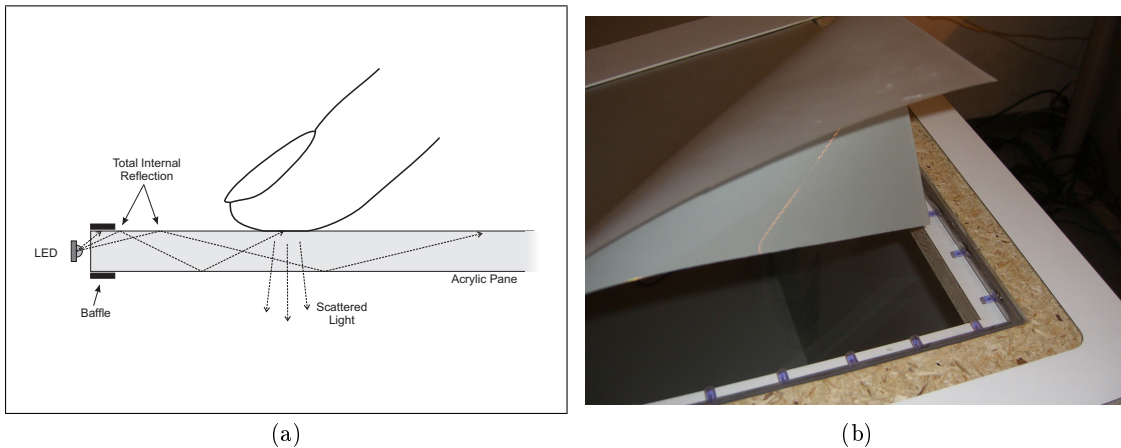


Figure 6.1: Frustrated Total Internal Reflection:

- (a) Schematic illustration of the principle (illustration according to [32])
- (b) photo of the LED-array and the overlaying foils

To make this construction not only a touch-sensitive surface but a touch-sensitive display, two layers of foil lay on the pane. The upper foil serves as projection screen for a digital projector mounted below the acrylic glass, and it provides interacting users with a pleasant surface feel. The lower foil makes sure that the display is still touch-sensitive: It is rather soft and creates, when pressed on the pane, diffusion spots of similar clarity as the ones created by skin. As a fortunate side effect, this foil also filters infrared light

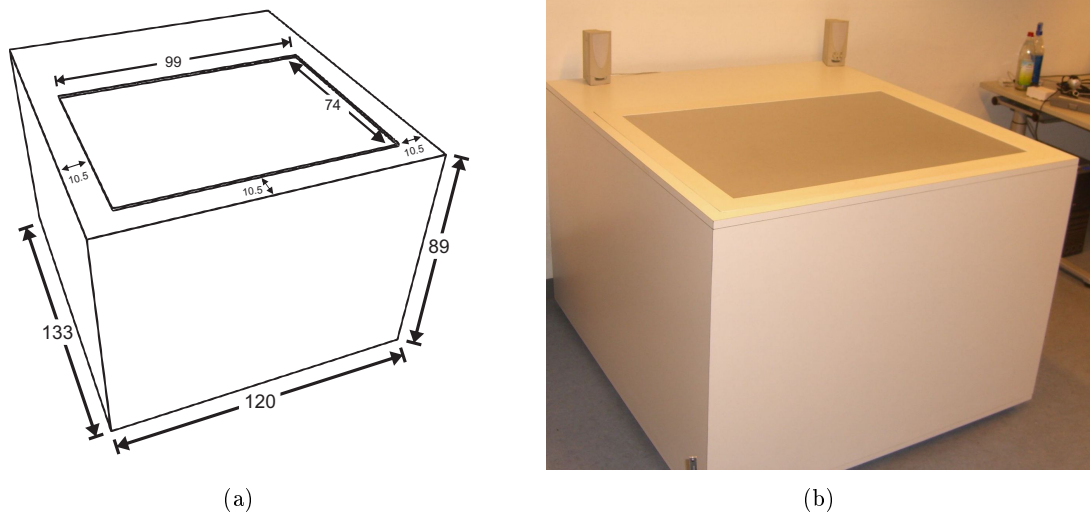


Figure 6.2: Measures (a) and photo (b) of the used tabletop device.

coming from the environment and enhances so the precision of the finger-tracking by reducing background noise. Figure 6.1(b) shows the described setup of acrylic glass, LEDs, and sheets of foil.

The tabletop display used for AudioPhield utilizes for the pressure-point recognition a gray-scale camera, which feeds its data to the computer via a FireWire connection. It delivers 60 frames per second at a resolution of 640×480 pixels. However, due to a barrel shaped distortion in the image caused by the lens, the optical resolution is not completely usable for tracking purposes. The display image is created by special projector, which is able to spread out the image at close distances thanks to specially shaped mirror mounted in front of the lens. Its native resolution is 1024×768 pixels. However, due to image adjustments handled not optically but by reshaping the image per software, and due to occlusions as a result of the projector's mounting position, the usable resolution adds up to about 950×700 . A closed box with a height of about 89 cm contains all the presented equipment. The touch-sensitive display measures about 99×74 cm and is located asymmetrically towards one of the short sides of the base area. A detailed scheme of these measures can be found in Figure 6.2(a). The box itself stands centrally in a room against a wall. Thus, users have to stand around the tabletop device and access the touch-area from three sides. This area's diagonal is larger than the average user's reach; so, users positioned at an edge of the table can not operate the whole interface. The rim around the sensitive field is with 10.5 cm broad enough for them to rest their arms there and support themselves while reaching out farther on the interface. See Figure 6.2(b) for a photo of the device.

The camera and the projector are connected to a usual desktop PC featuring a dual-core CPU and a middle-class consumer-grade graphics card. All computations necessary for AudioPhield are executed on this hardware. It is powerful enough to ensure that the frame-rate during usual interaction with AudioPhield is at all times above 30, and thus high enough to make the interface seem smooth and fluid.

6.2 General Software Design

This section is concerned with the used technologies and libraries as well as with AudioPhield's general architecture. Also, major components and classes are introduced.

6.2.1 Used technologies

AudioPhield is completely written in C#. There are various reasons for this choice, including type-safety, object-orientation, native compatibility to C/C++, performance, garbage-collection, and, last but not least, our extensive previous experience with the language. Also, Microsoft's excellent VisualStudio ([@13]) provides a convenient and effective development environment for this programming language.

Apart from this basic choice, AudioPhield relies on a number of other libraries, too. The following is a complete list of these libraries along with short descriptions of them:

XNA A lot of different meanings are combined by the term "XNA"³⁰ including web services, or communities. But at its heart it is a platform to develop games for Windows PCs and "Microsoft Xbox360" consoles. XNA provides a well structured API in form of a class hierarchy (for both, C# and VisualBasic), by which two- and three-dimensional graphics can be created and manipulated. On PCs, XNA is thereby essentially a wrapper: Most commands are mapped to DirectX-calls and executed by this high-performance graphic library. Thus, if the according graphic chips are available, most XNA-operations are hardware-accelerated [@15].

XNA GameStudio 2.0 The "XNA GameStudio 2.0" by Microsoft (short: GameStudio) is supposed to ease the entry into game development as much as possible and comprises therefore skeleton projects, which get tightly integrated into the VisualStudio, a rich library of utilities for game development, and a set of examples and tutorials [@14]. Its key value for AudioPhield is the automated handling of default tasks in a 3D program: A GameStudio-project automatically includes a powerful subroutine to load and maintain content in the video memory. Also, a central game-loop, which calls the user's update- and rendering-code, hides all timing issues.

Touchlib The open-source library "Touchlib" was especially designed for creating vision-based multi-touch surfaces [@21]. It handles thereby the complete image-processing and interpretation, from accessing the camera to tracking interaction points. To find and track touches – which appear as blobs in the camera image –, it preprocesses the image to separate the blobs from the background. Therefore, it transforms the image in several steps, like background reduction or contrast enhancement. Figure 6.3 demonstrates the effect of this process. Then, it identifies blobs and tracks them in the image stream coming from the camera. Finally, the centers of the blobs are calculated, transformed by a matrix-grid to cancel out lens distortions, and sent to listening programs in form of x/y-coordinates (normalized to [0;1]) and an identifying number for each blob. The previously mentioned tracking-losses or -interruptions happen here: The ID of a blob changes during its motion. AudioPhield uses currently a C# wrapper to start and access Touchlib directly. A earlier implemented approach used sockets for the communication and worked just as well – albeit with more overhead.

SmartSDK As it was initially unclear if the above presented table would be available for this thesis, AudioPhield was also targeted on a horizontally mounted SMART Board by SMART Technologies [@28]. Therefore, it incorporates the SmartSDK to access interaction events detected by the proprietary SMART software. While it is not advised because of the limit of only two simultaneous touch-points, AudioPhield may still be used on any SMART Board.

³⁰"XNA" is an arbitrarily chosen name and not an acronym. However, this description became a recursive acronym itself: "XNA's Not Acronymed".

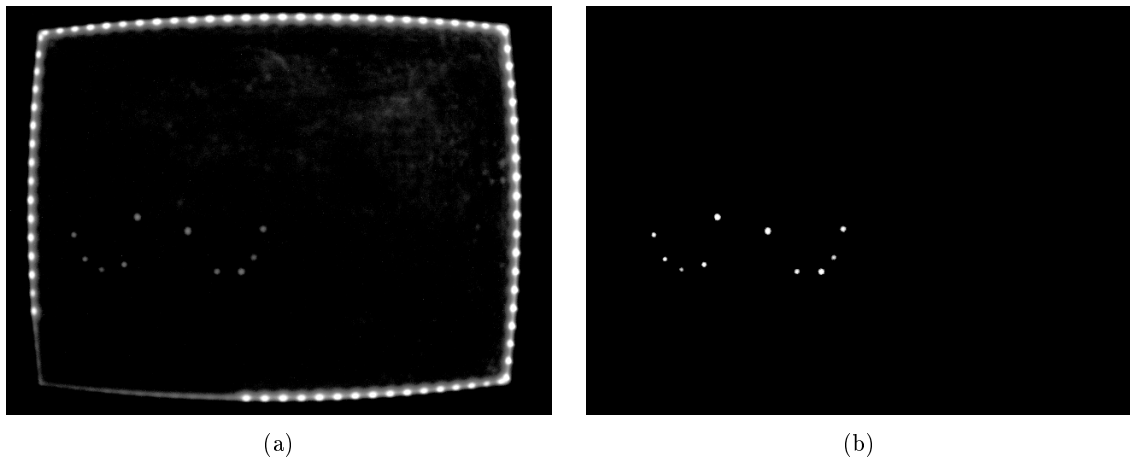


Figure 6.3: Image processing in Touchlib:
 (a) shows the image captured by the camera.
 (b) shows the same image after several image processing steps.

BASS The “BASS” library in version 2.4.1 by “un4seen developments” [31] handles most music issues in AudioPhield: It is responsible for decoding songs into floating-point PCM data³¹ for attribute extraction, and for playback of pieces of music during a running AudioPhield session. Among the many available music handling libraries it was chosen because it is very robust, flexible (a plug-in system enables the playback of virtually any file-format music is stored in), and easy to integrate into AudioPhield thanks to a C#-wrapper named “Bass.Net”. Also, BASS may be used at no charge in non-commercial projects. However, while it is able to read tags affixed to a music file acceptably, there is no function included to *write* tags.

AudioGenie The library “AudioGenie” [30] in version 1.0.4 fills the function gap of BASS. This freeware handles reading and writing of tags reasonably well and offers a very clean API³². The library comes as a single .dll-file with a C#-wrapper for convenient access. AudioGenie supports of the widespread digital music formats only .mp3, .wma and .ogg yet – but this may change in future versions. Since AudioPhield needs to store data inside a song’s tag, AudioPhield is currently also limited to these file-types.

6.2.2 General Architecture

AudioPhield consists at this time of roughly 85 classes and interfaces, which hold together close to 15,000 lines of code (comment and blank lines not included). So, it should be clear that an extensive description off all involved classes and found solutions is far beyond the scope of this thesis. We will therefore limit the following discussion of the system’s architecture to include only the most prominent classes and data-flows.

AudioPhield’s software architecture is designed to be very flexible and expandable. While most classes are loosely coupled towards each other, the following entities can be

³¹PCM stands for “Pulse-Code-Modulation”. It means essentially the transformation of an analog signal into binary data by sampling the signal. PCM data is thus the discrete approximated form of the original signal.

³²“API” is short for “Application Programming Interface”. As the name suggests, it is the interface a program or library offers to foreign programs for mutual access.

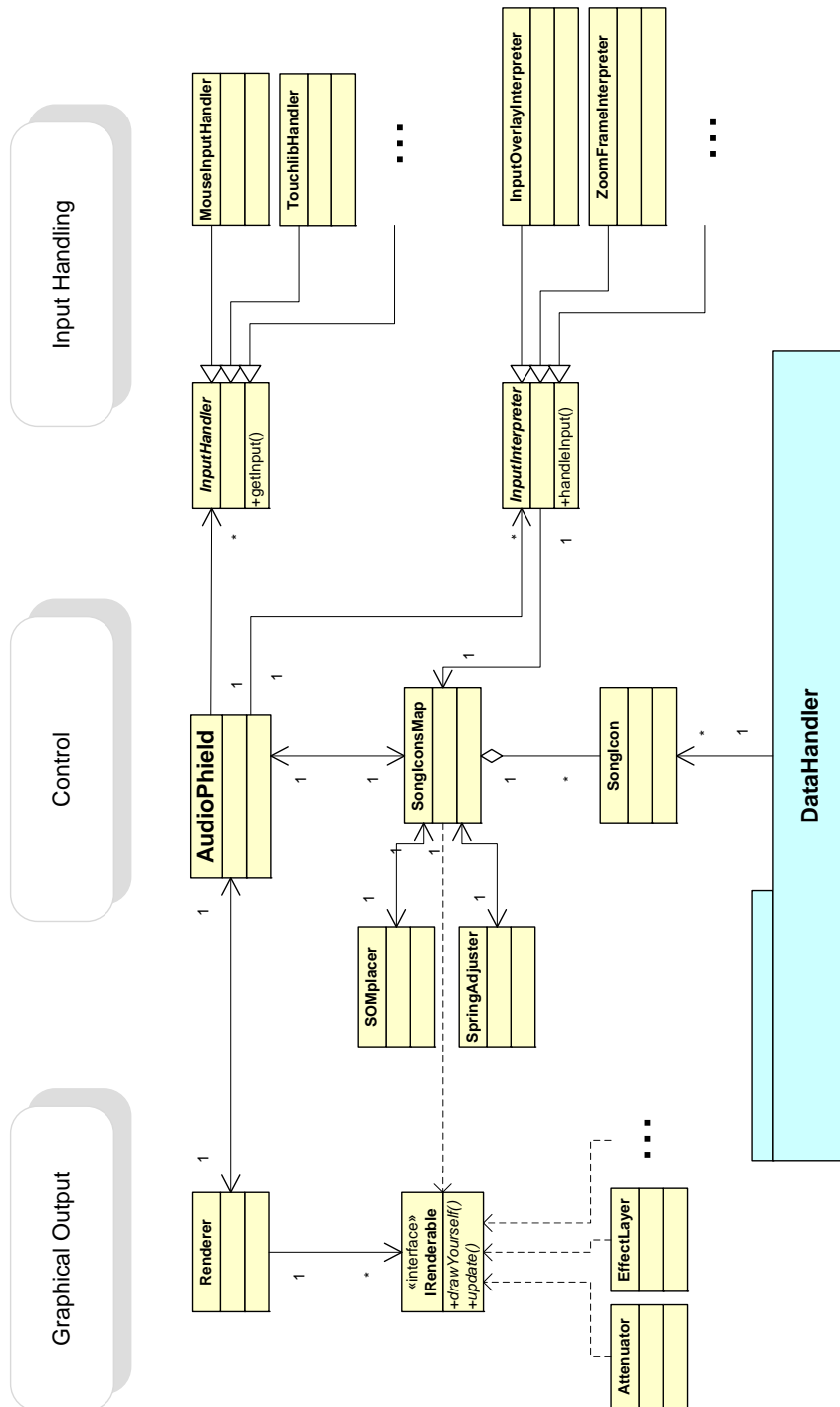


Figure 6.4: Strongly simplified class-diagram of AudioPhield's architecture in UML-notation.

considered AudioPhield's core. They are ordered roughly by their position in the class-diagram in Figure 6.4.

- **AudioPhield**: This class is the central control entity. It handles acquisition and dispatching of input-data. It is also the main factory to create the arrangement of the remaining classes.
- **Renderer**: The **Renderer** is responsible for general painting issues. It handles the access to the graphics hardware and contains update- and render-loop. In the render-loop, all **IRenderables** are called to "*drawYourself()*" in a sequence that determines their z-order.
- **IRenderable**: Every object that is to appear on the interface (or that needs to be called during the *update*-call that precedes the rendering) has to implement this interface and to register itself at the renderer. While some **IRenderables**' sole purpose is to enrich the user interface (e.g., the **EffectLayer**), others are not even painted, like the **Attenuator** that smooths the icon's movements.
- **SongIconsMap**: This class controls the positioning and drawing of the **SongIcons**. It holds and maintains the distortion grid (see below).
- **SOMplacer**: The **SOMplacer** implements the self-organizing map used to locate **SongIcons**. It is loosely coupled through an interface that contains only few methods and may thus easily be replaced by a different algorithm.
- **SpringAdjuster**: The spring-algorithm presented below is implemented in this class. Because the computations of the **SpringAdjuster** would damage the frame-rate noticeable, they run in a separate, low-priority thread.
- **SongIcon**: This class serves two purposes. It represents a song with all its attributes as storage class, and it handles the song's appearance as icon on the map.
- **InputHandler**: There are a number of classes that inherit from this abstract class. It is an **InputHandler**'s responsibility to collect input-events from a specific source and to convert them into the general **InputObject** (not shown in the diagram) used in AudioPhield. **InputHandlers** have to buffer all events until they are explicitly requested during the *update*-step. The above presented Touchlib and SmartSDK are wrapped in an **InputHandler** and further sources of input may easily be added the same way.
- **InputInterpreter**: Classes inheriting from the abstract **InputInterpreter** are sinks for **InputObjects** provided by the **InputHandlers**. They judge for themselves, if an input needs to be interpreted by them. The simple **InputOverlayInterpreter**, which visualizes subsequent **InputObjects** as lines, or the more complex **ZoomFrameInterpreter**, which completely implements the Zoom-Frame interaction-technique presented in 5.3.1, are examples of **InputInterpreters**.

As mentioned, this list only outlines the most prominent classes. For the sake of clarity, the complete factory-system, which builds up the different components of AudioPhield, or the song playback infrastructure are here omitted.

The input acquisition and processing is executed in every *update*-call, i.e., once per frame. It follows a simple principle illustrated by the following pseudo-code snippet:


```

1  for each handler in allInputHandlers
2    input = handler->getInput()
3    if input then
4      for each interpreter in allInputInterpreters
5        completelyInterpreted = interpreter->interpretInput(input)
6        if (completelyInterpreted) then break
7      else continue
8    else
9      continue

```

So, the handling is essentially just a nested loop in which all interpreters sorted by their z-order decide for themselves if an input should be interpreted and executed by them, and if the input is completely handled, i.e., if subsequent `InputInterpreters`, which lie possibly behind the current `Interpreter`, should also be allowed to handle the input.

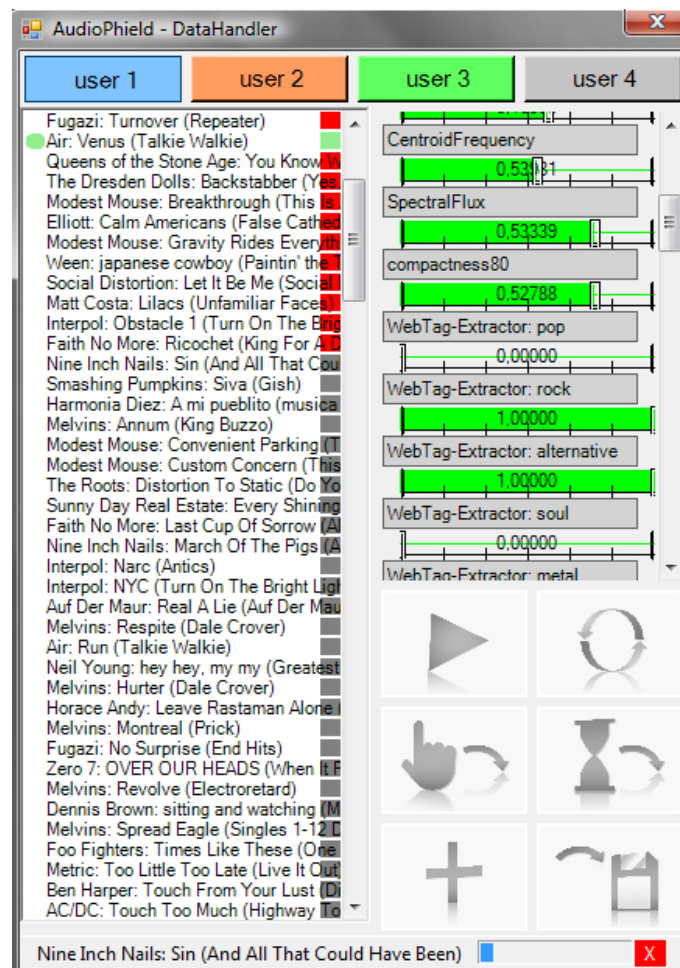


Figure 6.5: Screenshot of the DataHandler. The left list shows all songs assigned to a user. The upper right boxes display the extracted values; these may also be changed here manually. The lower right area contains the controls for ,e.g., addition of new songs, playback or automatic placement.

6.2.3 The DataHandler

One aspect shown in Figure 6.4 is still to be discussed: The DataHandler. This package is essentially a program of its own that operates almost completely isolated from the rest of AudioPhield – both parts of the system may even run solitary. The DataHandler enables users to select which songs are to be added to the similarity-based visualization on the tabletop. As soon as songs are associated with a user (the “+”-button in screenshot 6.5), the program decodes the song looking for tags. If a song was previously scanned by AudioPhield, custom tags are found in the song, which include information about play-count, the last position the according song-icon was placed, and a triple of extractor-name, extracted value and extraction accuracy for every extracted dimension (see below). So, the song can be processed by the **SOMplacer** and added to the field. If tags of this kind are not found, the song is considered unknown. Then, the extraction process starts automatically and – if the user wishes – the newly computed values are stored inside the songs themselves as tags. See Figure 6.5 for a screenshot of the DataHandler.

The DataHandler is, for the most part, the same framework for music information retrieval that we have presented in [75]. The complete architecture and most of the plug-in algorithms (as well as some new – as the web-extractors discussed below) from there are incorporated in this package. The only essential novelty is the new, multi-threaded GUI that hides the complete extraction process while offering about the same capabilities (apart from the additional buttons used to trigger song-placement). Therefore, we do not present this package here but refer interested readers to [75], where an exhaustive description of these matters can be found.

The coupling between DataHandler and the above presented part of AudioPhield is extremely loose: The communication is one-way only (from DataHandler to AudioPhield) and limited to simple calls like “*addSongAutoPlaced(...)*” or “*placeSongManually(...)*”. The only class both parts of AudioPhield use is the **SongIcon**.

6.3 Selected Implementation Aspects

As stated above, AudioPhield is by now too voluminous to be discussed in this thesis completely. However, some of the more interesting or unusual aspects of the implementation deserve to be discussed here. We have chosen therefore the used (and not used) algorithms to obtain information about songs necessary for the similarity depiction, our implementation of the Kohonen map, a spring-algorithm used to enhance the icon layout and finally the technique to create arbitrary free-form fisheyes.

6.3.1 Music Information Retrieval

The user’s experiences with AudioPhield depend heavily on the quality of the similarity-based icon placement. This again can only be as good as the data used to compute the similarity. In the following, we will discuss two fundamentally different approaches to obtain this necessary data. Their task, however is the same: Gather information that correlates with features used by humans to judge how a piece of music sounds, and express it by two single values normalized to the interval $[0; 1]$. The first value encodes the feature itself (e.g., ‘0.2’ for a fairly slow, ‘0.9’ for a really fast song when the feature is “speed”). The second value serves as metadata for the first one: It contains the estimated accuracy of the first value. So, if the ‘0.2’ in the example above is accompanied by a low accuracy of ‘0.1’, the algorithm means – to put it crudely – “I guess the song is fairly slow but actually I don’t know”. This meta-data can be as valuable as the data itself for the process of similarity computation as it prevents the harmful inclusion of faulty results.

Content-based Data

In our initial design, AudioPhield’s similarity estimations were supposed to rely entirely on data extracted from the music itself. Thus, many different algorithms were devised and implemented following the suggestions of various popular papers in the music information retrieval community – unfortunately with little success. This stems in part from the following problems with real-world music files:

- **Recording:** Pieces of music are to a good part shaped by the studio in which they were recorded. Differences in the strength of applied compressions or different equalizer modifications complicate, e.g., estimations of how “bass-dominated” a song may be.
- **Encodings:** Digital music is encoded by a range of programs. Virtually all commonly used codecs encode the music lossy by applying psychoacoustical insights, i.e., they omit sound information people usually are not able to hear. However, algorithms assuming a complete wave spectrum produce wrong results.
- **Segmentation:** Usual pieces of music tend to be not completely homogeneous. For example, refrains are in most cases more melodious than the strophes. Inconsistencies like that tend to confuse algorithms, which usually postulate uniform input.
- **Limited Scope:** Few algorithms are capable to create valid results for every kind of music – the bandwidth is just too large: Looking for energy peaks in lower frequency regions may be feasible for making suppositions about rhythmic patterns in pop music; for jazz, however, the results are worthless.

However, these problems hardly suffice to explain the the poor performance of our algorithms. It seems that the “right” way to teach computers to listen to music like humans do – or at least to identify the features people use to appraise pieces of music – are still to be found. We came to this realization because of the following reasons:

a) **Missing Encoding Robustness**

Many of the current approaches in Music Information-Retrieval use a set of (rather easy to compute) low-level attributes. After one of our attempts, which was to work similarly by training a neural network to classify songs according to a feature set recommended in [87], produced unrepeatabile results, we conducted an informal study on the influence of different encodings on selected low-level features³³. Therefore, we read out the uncompressed PCM data from CDs as a reference value. Then, we encoded the same data with different popular codecs, such as LAME [1] or libvorbis [33]. After that, the files were given to AudioPhield to extract the chosen features and save them for later evaluation. The diagrams in Figure 6.6 show an excerpt of the results. Obviously, the encoding has a tremendous impact on these features. Even the values extracted from the mp3-encoded songs differed sometimes greatly, although they were encoded by the same program with similar data rates.

Obviously, since AudioPhield uses BASS to decode and not the same codec as the ones used to encode the files, these artifacts could have been produced by BASS. Therefore, we stored the PCM data decoded by BASS and played it back to compare it with the original file from the CD. At least to our unschooled ear, there was no

³³These features were the average and variance of: ZeroCrossings, Spectral Centroid, Spectral Rolloff, Spectral Flux and LowEnergy ratio. For all computations were the same usual window widths and options used. The implementation of the used algorithms can be found in [75].

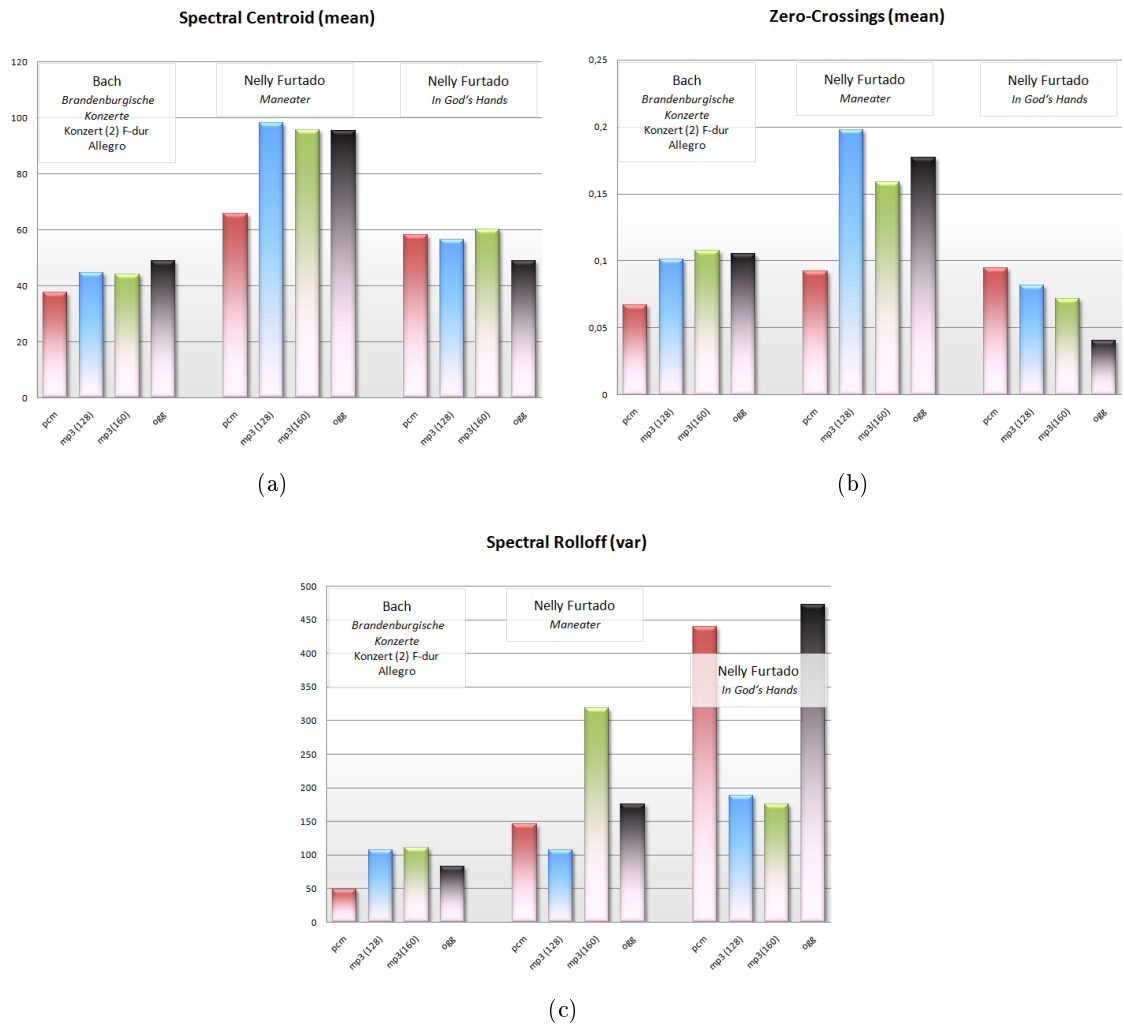


Figure 6.6: Illustration of the encoding robustness of selected features.

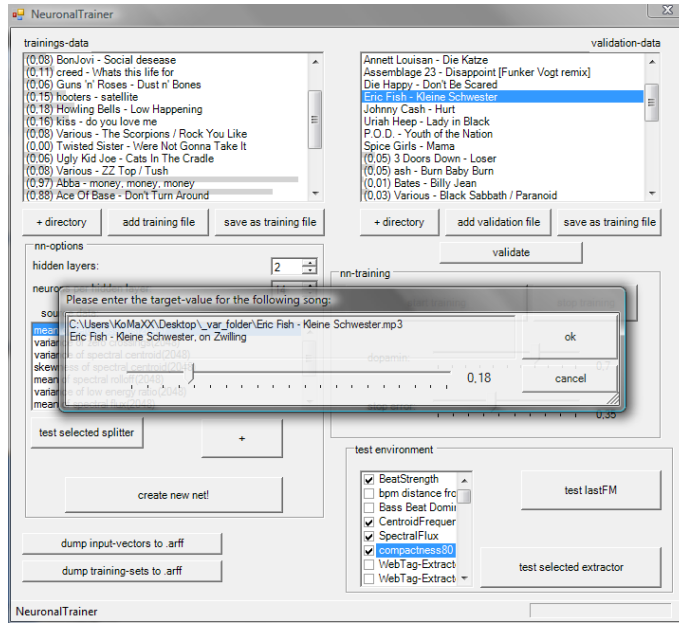


Figure 6.7: Screenshot of the music annotation tool.

perceivable difference – and certainly no difference of the same magnitude as the differences, e.g., in SpectralCentroid values.

So, the influence of the codec on various low-level features seems to be high enough to rival the influence of the data we actually want to obtain. Thus, without limiting AudioPhield to PCM data read directly from CD to achieve comparable circumstances, it is to be expected that algorithms based on low-level features of this kind deliver unsatisfying results.

b) Weak Statistical Correlations

Strong influences of the encoding robustness could still possibly become negligible when a *set* of features is combined to a statistical model that correlates with a meaningful feature of the music. We modified therefore a tool, which we previously created to train artificial neural networks, to let us quickly annotate a collection of music with “target” values, as, e.g., an estimation of a song’s mood or the dominance of the singing voice (See Figure 6.7 for a screenshot). The same program also executes all implemented feature-extractor algorithms and stores their results and the annotated value in a file-format that the data mining and machine learning framework “WEKA” [94] can read. Here, we tried to find correlations between the used feature sets and the high-level annotations – with little success. Seemingly, the low-level features we used (the same as above) are inapt as base for WEKA-classifiers like “NaiveBayes” or “ZeroR” to make statements about the cheerfulness, affinity to the genre “pop” and voice dominance (or at least to our estimations of these characteristics) for the arbitrarily selected 53 songs.

c) Results by Pohle *et al.*

Pohle *et al.* presented in [66] a detailed evaluation of the suitability of various combinations of feature extraction and machine learning algorithms for music estimations. The three examined feature sets stem from publications which are often used as references and cover up to 146 dimensions. The chosen machine learning algorithms, including decision trees, regression tests, and support vector machines, cover the standard algorithms used in MIR publications. These algorithms were trained with

a database of 834 manually annotated songs to classify them into perceptual categories like “perceived tempo”, “mood” or “complexity” on the basis of the different feature sets. Then, they compared the classification accuracy with the baseline, which is the accuracy of an algorithm that simply chooses always the most frequent class regardless of any features. The overall results showed that even the best tested combination was barely able to beat the baseline. Pohle *et al.* found for some classifications “no indication [...] that the most commonly used audio features are useful [...]”, and in most cases the overall classification accuracy was below the baseline.

It needs to be stressed that our informal, explorative investigations do not satisfy the requirements of a rigorous study and the used statistical methods are disputable. We are also not saying that it is impossible by principle to create algorithms which emulate human appreciation of music – albeit Aucouturier and Pachet indeed suspect an upper bound of the possibly achievable performance in [3]. However, these observations and the confirmation of our findings in [66] were discouraging enough to estimate that our approaches would not be futile in the foreseeable future. Thus, we discontinued further efforts in this field of research and included only the in the following discussed MIR-based algorithms.

One of the many dimensions in which pieces of music can be alike or differ is the perceived strength of the main beat. As Tzanetakis showed in [88], most people understand the feature “beat strength” roughly the same way and estimate it usually similar. To obtain information about the main beat of a song, we implemented an algorithm based on the “Beat Histogram”, a representation of a songs self-similarity, as introduced in [87]. To compute this histogram, the algorithm conducts the following 6 steps:

1. Decode the input-song to PCM data and cut out a part of 15 seconds after one third of the song’s duration. Only this part is used for further computations to improve computation speed and ignore falsifying influences of intros, which often differ significantly from the rest of the song.
2. Compute the “Short Time Fourier Transformation” (short: STFT) of the data. This transformation converts subsequent and overlapping “windows” of the wave-data from the time- into the frequency-domain. For a detailed introduction to the STFT and a detailed explanation of the computation process, see our previous work [75].
3. Combine the coefficients into three bands to separate rhythmic features in bass-, middle- and high-frequencies.
4. Subtract the average of a band from all values (“mean removal”) to enhance the signal for the autocorrelation.
5. Compute the autocorrelation of each band separately. This is done by the following function:

Autocorrelation y for periodicity k :

$$y[k] = \frac{1}{N} \sum_{n=1}^N x[n]x[n-k],$$

where $x[n]$ is the n th value in a band, and N a band’s cardinality.

The computation is thereby limited to the beats-per-minutes (short: bpm) found in non-exotic western music (40 to 130).

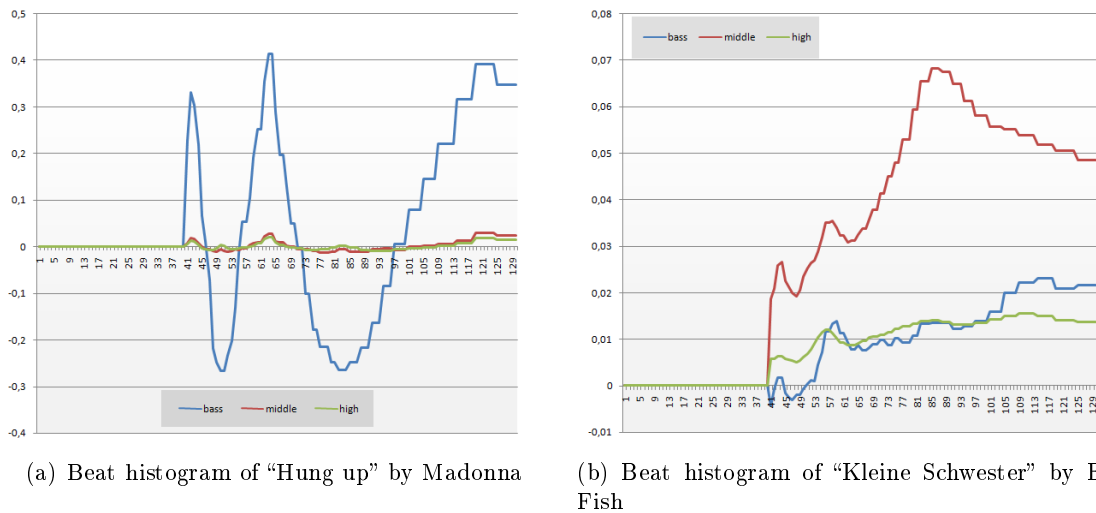


Figure 6.8: Two exemplary beat histograms.

6. Enhance the autocorrelation by removing integer multiples. For this, a copy of the result of the previous step is clipped to positive values, time-scaled by a factor of 2 (i.e., the new series of values is two times as long as the original one), and subtracted from the original series. Thus, the effect of integer multiples is reduced. In this way, periodicities like 60 and 120 bpm should influence each other less.

The result of these steps is illustrated in Figure 6.8 on the basis of Madonna’s “Hung up”, a danceable song with a driving beat, and an atmospheric ballad by Eric Fish, which is only accompanied by an acoustic guitar. These histograms are now the basis for actual feature-extraction algorithms. The first of these implemented in AudioPhield is the **BeatStrength**-extractor, which finds the highest peak in the bass-band and compares it to the average energy contained in the band. Also, a correction factor is applied to scale the resulting values to the required $[0; 1]$ interval. The second algorithm is called **BassBeatDominance**; it is supposed to estimate if the perceived beat is dominated by the bass band. Therefore, it adds the energy contained in the two highest peaks of the lowest band and divides it by the accumulated energy of the two highest peaks in the combined higher bands. Again, a scale-factor is multiplied to the result.

While these algorithms deliver understandable results in most cases, the extracted values are still not precise and robust enough to be of great value. The similarity-estimation process discussed below takes these features therefore only into minor account.

Folksonomy Data

As we discontinued our MIR approaches, a new source to obtain meaningful data about pieces of music from needed to be found. Luckily, the already several times mentioned web-service “last.fm” provides a simple API to access their database. This section will discuss briefly what folksonomy³⁴-data the service provides and how it is used in AudioPhield.

One of the functions, last.fm offers to its users is the possibility to annotate tracks, albums and artists with arbitrary tags, i.e., with short snippets of text. This function is quite popular and used for various purposes: There are personal tags like “seen live” (which is even one of the most popular tags), “favorite” (in various different spellings) or “one-mightbecomebarrenlisteningtothis” [6]. However, there are also many tags that describe

³⁴The term “folksonomy” is the fusion of “folk” and “taxonomy”. It is used as umbrella term for any system, in which casual users collaboratively classify or rate pieces of data.

the music in general. Among these are tags indicating the genre, mood, or instrumentation of a song, album or artist. This is also the kind of data that is valuable for AudioPhield – and, fortunately, it is accessible through “.getTags”-calls to last.fm’s API³⁵.

The following is a cutout from a typical dataset obtained this way (here for “The wizard” by Black Sabbath):

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <lfm status="ok">
3    <toptags artist="Black Sabbath" track="The Wizard">
4      <tag>
5        <name>heavy metal</name>
6        <count>98</count>
7        <url>www.last.fm/tag/heavy%20metal</url>
8      </tag>
9      <tag>
10       <name>metal</name>
11       <count>74</count>
12       <url>www.last.fm/tag/metal</url>
13     </tag>
14     <tag>
15       <name>classic rock</name>
16       <count>67</count>
17       <url>www.last.fm/tag/classic%20rock</url>
18     </tag>
19     <tag>
20       <name>hard rock</name>
21       <count>50</count>
22       <url>www.last.fm/tag/hard%20rock</url>
23     </tag>
24     ...
29     <tag>
30       <name>70s</name>
31       <count>25</count>
32       <url>www.last.fm/tag/70s</url>
33     </tag>
34     ...
79     <tag>
80       <name>guitar</name>
81       <count>15</count>
82       <url>www.last.fm/tag/guitar</url>
83     </tag>
84     ...

```

This cutout only shows the first few lines of the data sent by last.fm. The original xml-file goes on for 450 more lines with more tags associated with the song. Obviously, this tag-data can be of great value for AudioPhield: Even if the users could not agree on the question, to which genre a song belongs, to know which genres possibly fit is even more meaningful than a clear classification into a single genre. Also, tags like “70s” or “guitar” give insights about issues that even our most ambitious content-based approaches could not target.

However, this information is still to be transformed into data usable in AudioPhield. The fact that the tags can be attached arbitrarily without supervision, causes some problems. There are large amounts of tags that are just spelled differently (e.g., “hip-hop” and “hiphop”), effectively used as synonyms (e.g., “hip-hop” and “rap”) or at least very close

³⁵A complete overview of this API can be found in [8]. It contains at the moment no possibility to obtain data that would be even more valuable for AudioPhield: The precise similarity data last.fm uses internally.

(e.g., “doom metal” and “death-doom metal”). On the other hand, there is also a lot of noise in the form of outright wrong tags. Most of these wrong tags appear on the bottom of the list and have a small “count”-value, and are thus easy to delete by just applying a threshold – ‘4’ seemed to work well in our experience. Unfortunately, the tag-list for artists does not contain a “count”-value of this kind. Furthermore, erroneous tags also appear closer to the top of the lists occasionally. To cope with these difficulties, we developed a system of extractors based on the list of the most popular tags. Every extractor represents thereby a rather broad feature and examines the available tags, if indicators are present that suggest a song’s belonging to their class. To do this, the extractors access a list of manually set rules to recognize a tag’s meaning. The following cutout shows the xml-file that holds all these rules (see Appendix A for the complete file):

```

...
127 <web_extractor significance="10">
128   <contains>
129     <any>electro</any>
130     <word>house</word>
131     <match>dance</match>
132     <word>eurodance</word>
133     <any>techno</any>
134     <word>rave</word>
135     <word>trance</word>
136   </contains>
137
138   <contras>
139     <match>rock</match>
140     <any>metal</any>
141     <match>hip_hop</match>
142     <match>hip-hop</match>
143   </contras>
144 </web_extractor>
...

```

Each such **web_extractor** entity defines one extractor. For each extractor a list of search-terms is defined, which may either indicate that the song fits this extractor (inside the **contains**-environment) or not (inside the **contras**-environment). These search-terms must be applied with different rigor: A **match**-term has to equal the tag exactly; if a term is marked as **word**, it must be separated by other words in the same tag by spaces; the **any**-rule is the softest as here the term just needs to appear anywhere in the tag, even somewhere inside a word.

So, an extractor knows now for the presence of which tags to look and how to do it. Therefore, it is time to compute a single estimation value and an accompanying accuracy value out of these indicators, as required by the similarity computation. We developed therefore an algorithm, which first counts all appearances of indicators and contra-indicators in the tags assigned to the song and in the tags assigned to the artist and the album. Then, the program essentially applies the decision tree depicted in Figure 6.9. So, for some default cases, no actual computation is executed. The accuracy value for cases where only indicators for albums or artists could be found, is lower than when tags describe the track itself, because a song may well not be typical for an album or an artist. Also, bands tend to change their style over time. Thus, only track-tags can cause an accuracy of 1. The computation formula the figure refers to in the lower right is quite simple. It computes the estimation-value r and the accuracy-value a based on the amount of indicators i_p and contra-indicators i_c :

$$r = |i_p| - (0.5 * |i_c|), \quad a = 1 - \frac{0.5 * |i_c|}{|i_p|}$$

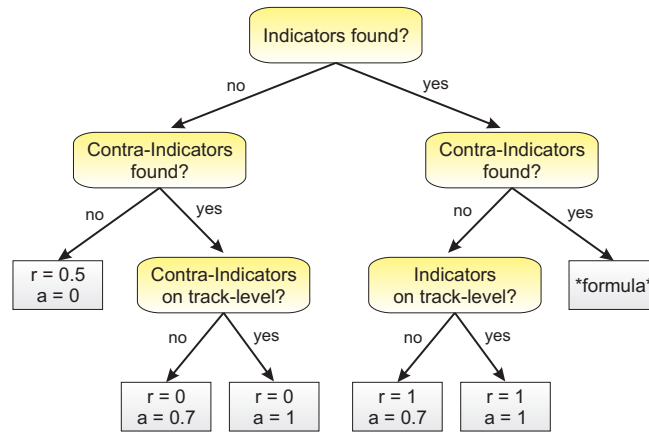


Figure 6.9: Decision tree of tag extractors.

The results are afterwards, if necessary, clipped to $[0; 1]$. The 0.5 factors strengthen the influence of pro-indicators against their counterparts to avoid situations where tags would cancel each other out: When about equal amounts of indicators and counter-indicators are found, then perhaps *both* are true. Consider for example a duet. This song may have “male voice”-tags and “female voice”-tags, which normally contradict each other. Without the factor, both extractors would return zero as accuracy and thus not be implied in similarity computations, although they could contribute valuable information.

The approach to use information from tags derived from a folksonomy has some clear advantages: The resulting values correlate to very high-level features of the music in terms which are natural to users. Furthermore, some tags, as, e.g., “guitar”, are usually only attached to a song, when the guitar is unusual dominant or otherwise peculiar in the song. So, these tags contain also an estimation of how important a feature is for the considered piece of music. On the other hand, however, satisfying results are limited to the genres and artists, a sufficient part of the community is interested in. Otherwise, there are usually just too few tags to deduce a detailed estimation. So, because the users of last.fm at large seem uninterested in reggae, even prominent artists as Bob Marley are only sparsely tagged – and classical music is practically not tagged. But even if an overwhelming amount of tags is available, the deducted information stays essentially binary: Either a tag is present or not. So, the extractors can only decide if a song is considered, e.g., melancholic – but not how melancholic on an absolute scale or compared to other songs. Because of this, the profile, which all tag-based extractors combined can create, remains rather imprecise. A more in-depth discussion of data deduction from folksonomies can be found in [41].

6.3.2 The SOM

Chapter 2.1 already introduced self-organizing maps as means to locate data entities after their similarity. Then, in section 5.1.3 we further discussed its usage in AudioPhield. Now, this chapter shall demonstrate how the SOM in AudioPhield works and what its peculiarities are.

As stated before, Kohonen maps, as SOMs are also called, belong to class of unsupervised learning neural networks. This means that there are no correct results, which the net is supposed to learn and use for classifications. Instead, the network is just provided with input-data and adapts itself by using the following technique:

The SOM is built up as a rectangular grid of “neuron” called nodes. Each of these

neurons represents a – initially random – point in the high-dimensional space of the input data. The training algorithm now seeks for any item in the input set the neuron whose point in this space is closer to the point specified by the input object than those of all other neurons. Then, this neuron, as well as some nodes in its proximity, are adjusted according to the values of the item. In other words, the nodes in the SOM move towards the point in the feature-space. Because the influence of new items decreases over time, at some point a static status is reached. Figure 6.10 demonstrates the principle with a fictive Kohonen map trained to classify colors.

The SOM in AudioPhield follows this scheme largely. It is also a regular grid of nodes that covers the whole surface, albeit with 80 nodes per axis finer than usual implementations (see, e.g., [39]). This number of nodes was chosen because it is high enough to enable fine nuances in close proximities, and low enough to make searches and modifications computable in reasonable time³⁶. The input data for this SOM consists of the feature vectors attached to individual songs. Since there are 45 different extractors, these feature vectors are 45-dimensional. So, each neuron in the SOM also holds a vector of this dimensionality.

Like many other SOM implementations, the Kohonen map in our system uses “online” training instead of “batch” training, i.e., each neuron is adjusted as soon as it was determined to match an input vector best. In “batch” training, the algorithm first finds the closest neuron to each input item and then updates the neurons collectively. Also, the training process in AudioPhield is limited to a single epoch. That means that every input object is only applied once to the SOM and then immediately placed close to the best matching neuron – but not exactly on it to avoid that the icons are placed on top of each other. In this way, we get immediate results, and the SOM needs not to be recomputed completely when songs are added or removed from the input set. As a downside, the risk arises that the formerly closest neuron “moves away” from an already placed song-icon when later added items alter the same area. Due to the pre-imprinting (see below) this happens rarely in our experience.

For the usually slowly diminishing adjustment-strength and -reach, we use a very simple alternative: In AudioPhield both values stay always the same. The reason for this is the pre-imprinting as described below. The high influence of items added to the SOM early in many implementations shall ensure that the SOM is partitioned into few major clusters instead of many, very similar, small clusters. This risk is non-existent in our approach because the pre-imprinting circumvents it effectively. Which neurons are affected by an adjustment, and to what extent, is determined by computing a certain reach factor divided by the squared Euclidean distance.

To compute the distance between points in the feature space, it is important to consider that this space is by no means Euclidean, and its “dimensions” are hardly linearly independent; after all, they encode to what extent music listeners deem a fuzzy defined term appropriate to a song. Therefore, the common Euclidean distance (see footnote on page 6 for the formula) is not the best possibility to compute distances in the feature space, as it regards all dimensions as equally important. Also, it can not take inaccuracies into account. Therefore, based on the Manhattan distance d_m in the n -dimensional feature space

$$d_m(\vec{x}, \vec{y}) = \sum_{i=1}^n |x_i - y_i|, \text{ with } \vec{x}, \vec{y} \in F^n$$

³⁶ According to Mörchen’s taxonomy in [58], AudioPhield’s SOM is thus technically an “Emergent SOM”, or ESOM.

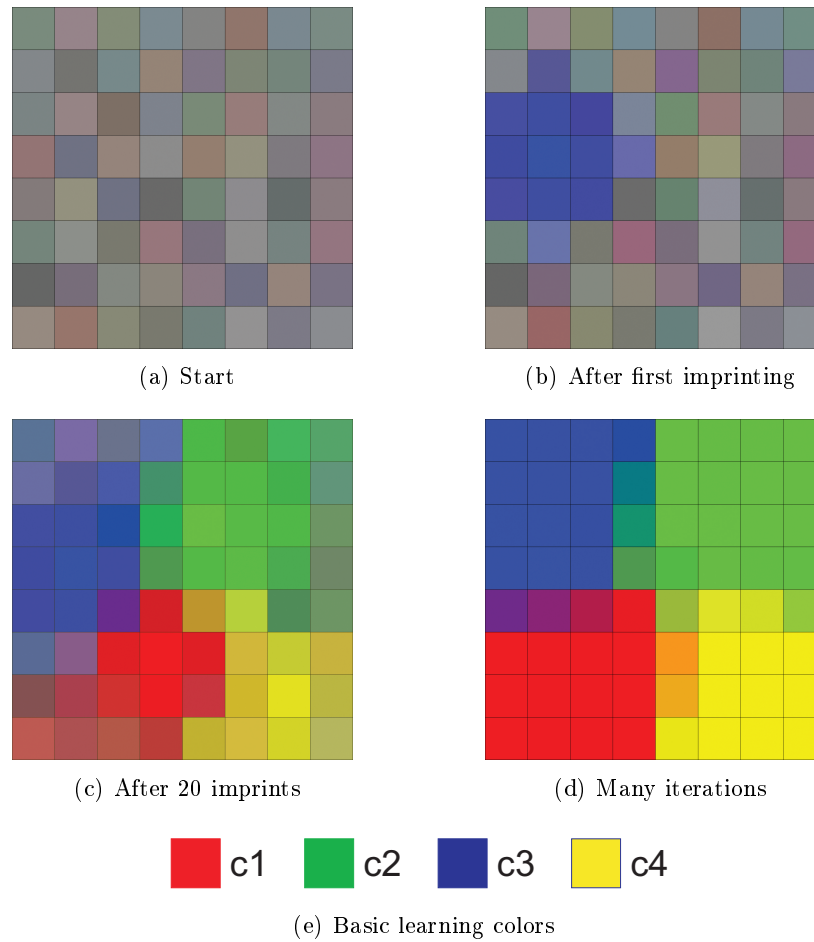


Figure 6.10: Illustration of the learning process of a 8x8 SOM.

Each square represents one node of the SOM. The data space has three dimensions, which are depicted here as the color channels red, green and blue.

(a) shows the SOM before the learning process started. The nodes hold random weight vectors close to the neutral gray to enhance the visibility of the learning process.

(b) and (c) show the result after the SOM was fed one and 20 random colors similar (but not necessarily identical) to the ones depicted in (e).

(d) shows the final state after many iterations. Note the clear separation between the colors.

we devised the “double-weighted Manhattan distance” d_{wwm} :

$$d_{wwm}(\vec{x}, \vec{y}) = \sum_{i=1}^n (w_{d_i} w_{a_i} |x_i - y_i|), \text{ with } \vec{x}, \vec{y} \in F^n.$$

We chose a modified version of the Manhattan distance because it makes no assumptions about connections between the different dimensions. The first applied weight factor w_d is a n -dimensional vector that describes the relative importance of a dimension, and thus, a feature. In this way, we can control the influence a feature has on the overall placement process. E.g., if two songs have the same “cheerfulness”-value but diametrically opposed “industrial”-values, then the latter should be of more importance than the former – because, even if both songs are cheerful, they will most likely still sound very different. While the first weight w_d is defined globally for all input items, the second weight w_a , which is also a n -dimensional vector, is defined for each item individually: w_a contains the accuracy value that each extractor computed besides the actual value. With this modification factor, the influence of potentially erroneous results can be weakened. If, e.g., there were no tags found, which could indicate if a song features a female singer or not, the corresponding extractor returned an accuracy of zero. So, this dimension is utterly ignored for the distance computation process, and the placement relies solely on other features with higher accuracies. Note that this distance metric makes it impossible to compare distances of different song icons from a certain neuron: If for one song were significantly more tags found, then its distance will generally be greater because less dimensions are ignored by the computation. However, since we only need to compare the differences between nodes and song icons, this peculiarity is irrelevant here. The accuracy values serve also a second purpose: When neurons are adjusted according to the features of a song, the strength of the adjustment depends on this also on this accuracy. This way, the influence of erroneous features is further weakened.

As introduced in 5.1.3, the SOM in Audiohield needs to be pre-imprinted for various reasons. So, while the vectors held by the neurons are still initialized randomly, a fixed set of feature-vectors is applied to the Kohonen map at defined positions. To ensure that the SOM is well partitioned, this preliminary training step is executed with significantly stronger impact and reach than the later, normal training. Since this imprinting is supposed to only define the SOM’s layout at large and not in any detail, the used feature-sets are only defined in the most influential dimensions, like, e.g., some of the genre features. Thus, while it is ensured that music belonging to the same major class of music, there still can emerge local clusters according to features with less impact, as instrumentation or decade of publication. The necessary feature-sets are manually refined vectors from automatically classified songs. A special tool called “SOMpregnator” was integrated into the DataHandler to allow the arbitrary placement of these vectors, and thus, the creation of a pre-imprint-map. Figure 6.11 shows a screenshot of this tool. The displayed layout was also used for the user study in the next chapter. It is, however, created absolutely arbitrary and the only justification of this layout is that it seemed sensible to us. Further studies are necessary to find a more universally valid layout. See Figure 6.13(a) for a screenshot of automatically placed songs.

6.3.3 Spring-algorithm

The above mentioned method during the item placement to avoid song icons being located on top of each other is functional but seems insufficient in some cases. Therefore, we included a force-directed algorithm that refines the layout to reduce overlaps. The principle of this algorithm mimics easy physics: Song-icons are considered as carrying equal electric charges, and thus exerting repulsive forces on each other. At the same time, they are

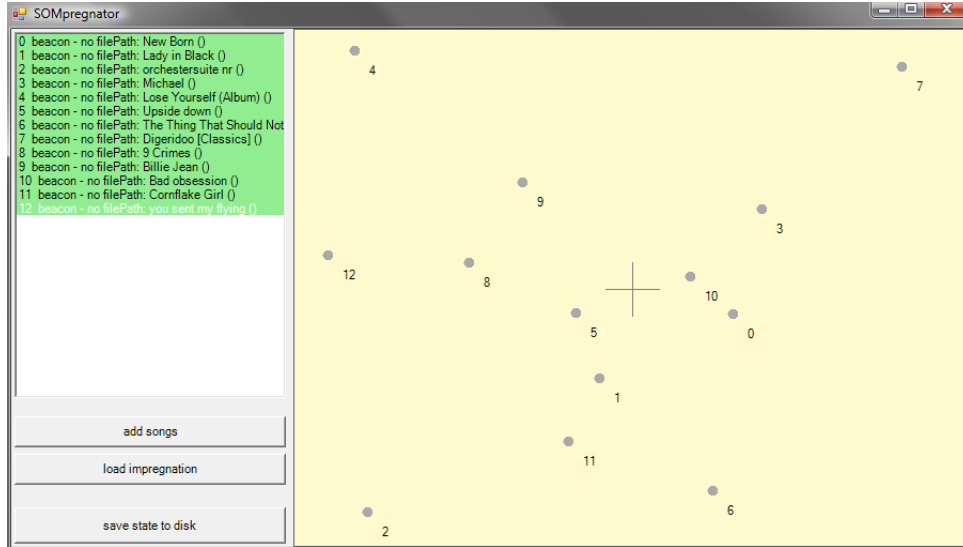
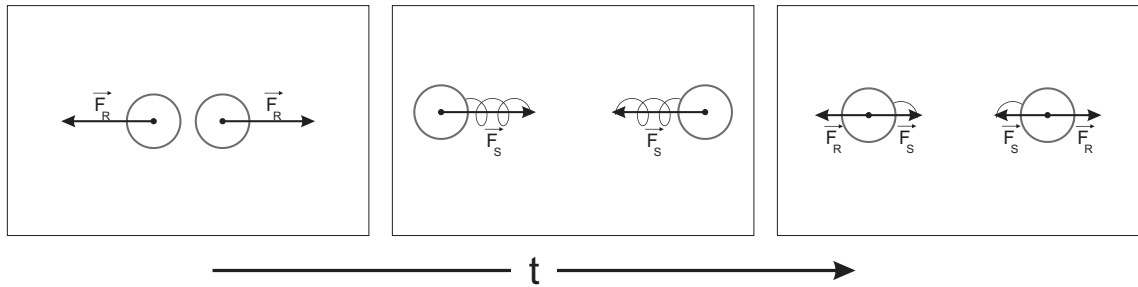


Figure 6.11: Screenshot of the SOMpregnator.

Figure 6.12: Illustration of the forces exerted on two items. \vec{F}_S names the force by the spring pulling to the original location. \vec{F}_R names the repulsing force between icons.

connected to their original position by a spring. As soon as the icons are then released, they interfere with each other and seek positions where all forces applied to them equal each other out. Figure 6.12 demonstrates the principle for two items.

The algorithm retraces this procedure in discrete time-steps. So, the system is not actively looking for the best possible layout but just computes simple parallelograms of forces. The following pseudo-code illustrates the procedure translated into software:

```

1 while(notAborted)
2   distanceMatrix = computeDistanceMatrix(allIcons)
3   foreach icon in allIcons
4     forceVector = SPRING_STRENGTH * icon->vectorToOrigin()
5     foreach otherIcon in allIcons
6       repelAmount = REPULSION_STRENGTH /
7         distanceMatrix->distanceBetween(icon, otherIcon)
8       repelDirection = otherIcon->position - icon->position
9       forceVector += repelAmount * repelDirection
10    newPosition = icon->currentPosition + ACTIVITY * forceVector
11    icon->setPosition(newPosition)

```

The effect of this non-terminating algorithm depends obviously on the three factors `SPRING_STRENGTH`, `REPULSION_STRENGTH` and `ACTIVITY`. The first two essentially specify

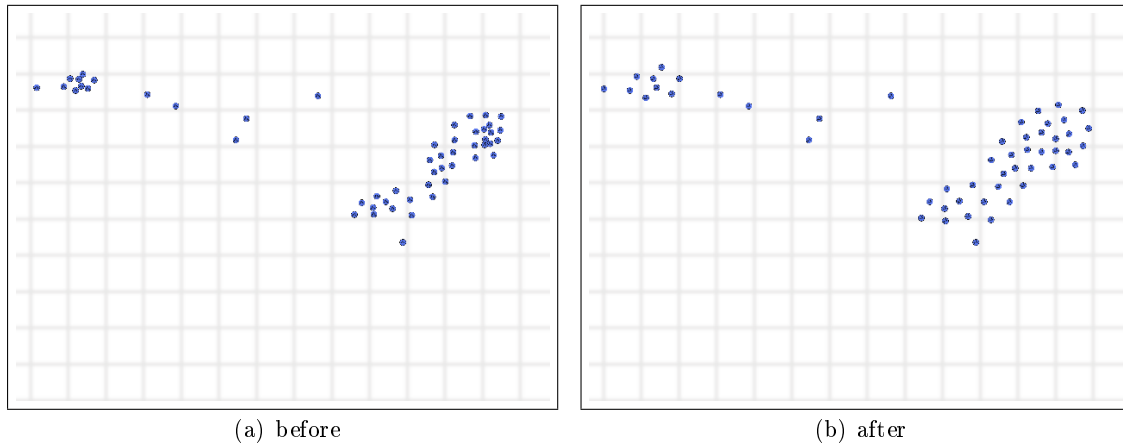


Figure 6.13: Comparison of the placement before and after the spring-algorithm is applied.

the ratio between the force exerted by the spring and the repulsion force. So, a higher `SPRING_STRENGTH`-value causes the icons to stay closer to their original position. The `ACTIVITY`-value effectively regulates the speed of the algorithm: High values (close to 1) cause rapid adjustment but also cause the icons to “flicker”. Low values, on the other hand, enable the forces to come closer to the desired equilibrium – but quite some time may pass until results are visible. To combine the advantages of both alternatives, the algorithm starts with a high `ACTIVITY` and decreases the value once after every pass of the main-loop by a certain percentage. See Figure 6.13 for a demonstration of the algorithm’s effect. It should be clearly visible that the icons in the second picture (b) are significantly further separated from each other. The figure shows also one downside of the spring-algorithm: The very dense cluster on the right in panel (a) has essentially vanished in panel (b). Thus, information was lost. However, even if this problem might not be avoidable at all, we believe that it could be at least reducible after further development – e.g., by including similarity matrices in the calculations as Hlavac demonstrated in [39].

6.3.4 Free-form Fisheyes

As chapter 5.2.1 on page 28 discussed, AudioPhield shall offer fisheye views to magnify areas of the field. Therefore, we initially implemented the generic fisheye-function as introduced by Sarkar and Brown in [73]. We could thereby not simply apply the fisheye lens on the visualization as a whole as if it was simply an image: After all, the icons are not static but rotate about their center, and the fisheye is supposed to trigger the display of additional information in the radial texts. These effects would be impossible to achieve if the fisheye was just moving pixels. So, the system calculates the distortion and magnification for each icon separately.

Therefore, it computes a distortion factor f and a magnification factor m in dependence of the distance x from the center of a fisheye lens for every icon, whose distance from this center is smaller than the maximum range x_{max} of the lens, according to these formulas:

$$f(x) = \frac{(d+1) \frac{x}{x_{max}}}{(d \frac{|x|}{x_{max}} + 1)} \quad , \text{ and } \quad m(x) = \frac{(d+1)}{(d \frac{|x|}{x_{max}} + 1)^2}$$

Here, d is the *distortion factor*, which determines the steepness of the distortion curve and the maximum magnification³⁷. Figure 6.14 shows the plots of both functions for $d = 3$ and

³⁷Both statements are actually equivalent, as the magnification function is the first derivative of the transformation function.

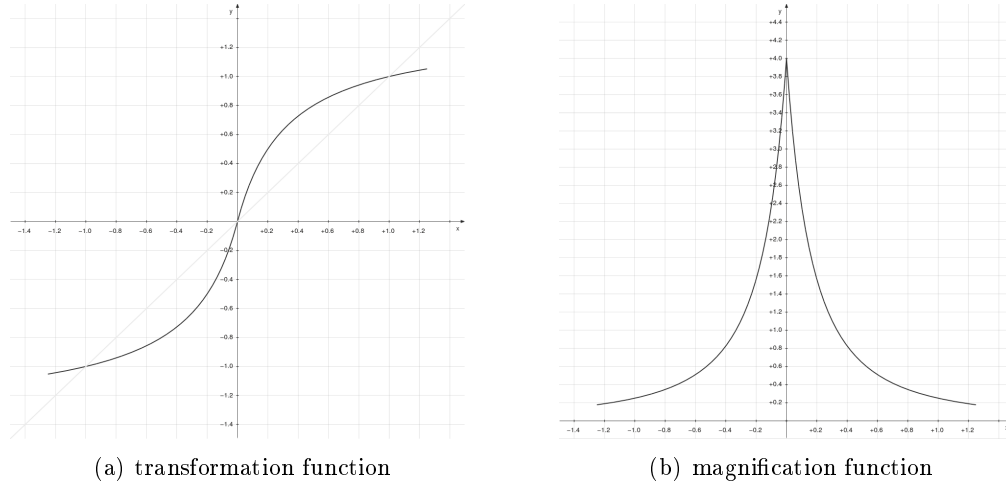


Figure 6.14: Plot of the graphs of the transformation function and magnification function of the default fisheye view for $d = 3$ and $x_{max} = 1$.

$x_{max} = 1$. The so generated value f is then simply multiplied on the delta-vector between the center of the fisheye lens and the icon to obtain the transformed position of the latter. Accordingly, the magnification factor m is multiplied with the basic size of the icon to get the new, magnified size.

Figure 6.14(b) reveals a problem of these functions: The magnification reaches very high values in the center before it decreases rapidly. Also, the distortion in the middle of the fisheye is very strong for a small area. As a consequence, icons need to be very close to the center to achieve the necessary magnification to display both radial texts (compare section 5.2.2, page 32). Worse, tiny movements of the fisheye lens are enough to cause icons in the middle of a focus area to move big distances. So, it is virtually impossible to get song icons to the very center of a lens and examine them soundly, and the whole interface seems rather nervous.

Our approach to solve this problem was to introduce a “plateau”. This is an area in the center of the fisheye distortion with a constant magnification and linear distortion. Here are the formulae of the in this way modified functions $f(x)$ and $m(x)$:

$$f(x) = \begin{cases} d_p * x & : x < x_p \\ \frac{(d+1) \frac{x}{x_{max}}}{(d \frac{|x|}{x_{max}} + 1)} & : x \geq x_p \end{cases}, \text{ and } m(x) = \begin{cases} \frac{(d+1)}{(d \frac{x_p}{x_{max}} + 1)^2} & : x < x_p \\ \frac{(d+1)}{(d \frac{|x|}{x_{max}} + 1)^2} & : x \geq x_p \end{cases},$$

where x_p is the range of the plateau, and d_p is the constant translation-factor for the linear distortion in the plateau. Figure 6.15 shows the corresponding graphs. The magnification in the plateau is constant and continues the function beyond x_p to avoid magnification jumps. However, since the transformation function is not continuous, here jumps exist. These occur when the fisheye lens is moved in such a way that an icon, which was previously further than x_p from the lens’ center, is now closer than x_p – i.e., when a song icon enters the plateau. Then, the icon “snaps” upon the plateau. In this instant, it does not move smoothly like before but jumps from one location to the other. While this may sound like undesired behavior, this approach is clearly superior to the default fisheye view when adequate values for x_p and d_p are chosen. Now, the user can deliberately use the plateau to “catch” song icons. In contrast to before, they do not get more uncontrollable the closer they get to the fisheye’s center but move slower and are thus easier to examine and play. Also, the jump discontinuity in the transformation function causes a ring around the

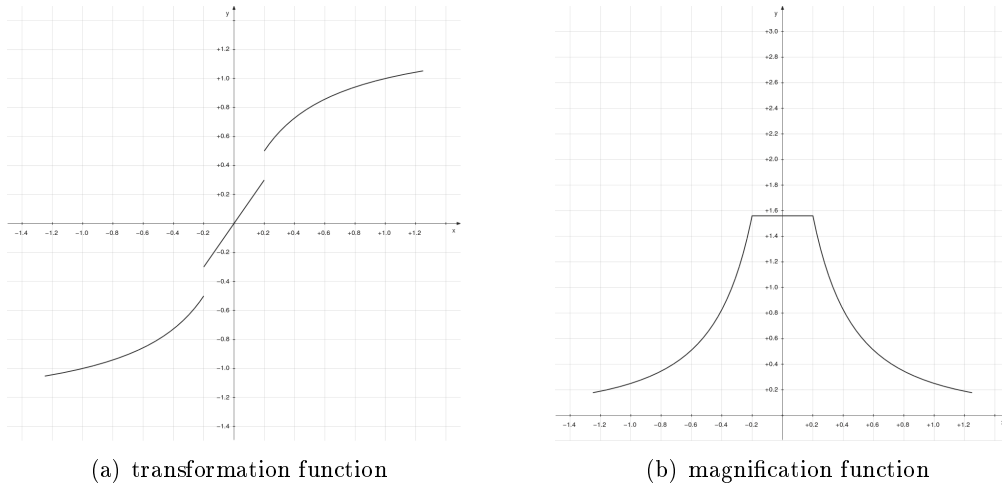


Figure 6.15: Plot of the graphs of the transformation function and magnification function of the fisheye view with plateau for $d = 3$, $x_{max} = 1$, $x_p = 0.2$ and $d_p = 1.5$.

plateau, where no icons appear. This ring is useful as it clears space for the additional space requirements of the radial texts.

However, this approach of directly applying fisheye distortions and magnifications to icons does not meet all of our requirements. For once, it does not scale well with the number of icons in the visualization because the (rather costly) transformation and magnification functions are to be recalculated for each icon and each frame. This problem multiplies with the number of fisheyes used simultaneously – so, it scales even worse with the number of users. In the case of multiple fisheyes the computations when these overlap would be a challenge, too. Furthermore, the principle is rather limiting: There is no way to create focus areas that are not perfectly circular. Concepts like the original SoapSpots or CookieDough (see chapter 5.3.1) are thus not possible.

Consequently, we developed a new approach based on a separate distortion layer: Therefore, a finely woven, rectangular grid of nodes covers the whole interface. These nodes are so-called “distortion nodes”. Each of them stores a two-dimensional translation vector and a magnification value. Now, the functions that are to create areas of focus do not influence the depiction of the song icons themselves but only the values stored in the distortion grid. In this way, there is a clear upper bound of the calculations necessary in one frame because the number of distortion nodes stays constant regardless of the amount of song icons or fisheye lenses in the visualization. The position and zoom of an icon is computed by simply combining the values of the nearest distortion nodes linearly and applying the result of this operation to the icon’s size and position. This operation is significantly less expensive than the computation of the fisheye functions – and may furthermore be executed hardware-accelerated by the dedicated graphic processor. Thus, this approach scales better with the number of icons. See Figure 6.16 for an illustration.

Also, the distortion and magnification stored in each node may be freely and separately set. This feature enables the implementation of arbitrary methods to create focus areas, as, e.g., the CookieDough interface, as described on page 42, where icons may just be moved without changing their magnification. Also, there is no need anymore for strictly circular focus areas – thus free-form fisheyes are possible.

The distortion grid has another peculiarity: The program parts, which create and maintain focus areas, are just allowed to set *target*-values for the distortion vector and magnification – not the current values. Another class, the already mentioned **Attenuator**, has the purpose to adjust the current values towards the set targets. This happens

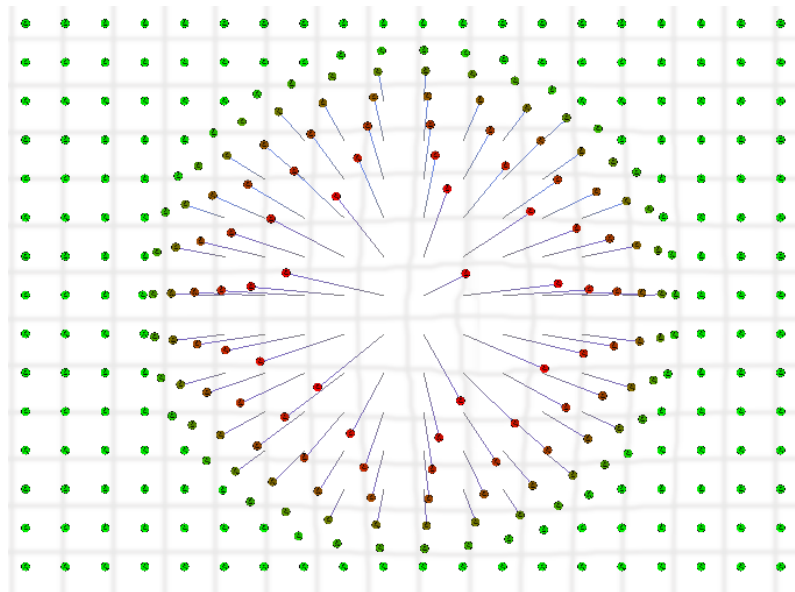


Figure 6.16: Illustration of a low-resolution distortion grid. The lines are supposed to visualize the distortion vector of a grid-node. Its magnification value is indicated by the color-coding of the small disks: the higher the red-value, the higher the stored magnification.

asymmetrical depending on the change direction: If the magnification and distortion is to be increased, the adaption occurs usually significantly more quickly than in the other direction. The exact speeds are determined by the active class to handle focus areas. Thus, concepts like the original SoapSpots, where areas of focus slowly return to their undistorted state, are possible. Furthermore, the whole interface seems to react smoother because all position and magnification changes occur gracefully animated. This is especially notable and valuable when song icons snap into a fisheye lens' plateau or out of it because the abrupt jump from the original concept is now a swift movement to the new location. Thus, the effect of the lens is better comprehensible. Figure 6.17 illustrates the distortion grid in combination with a very low set decrease-attenuation.

Now, we are confident that the most interesting features and solutions implemented in AudioPhield were made clear. The next chapter has to show if this implementation matches the design of the previous chapter sufficiently.

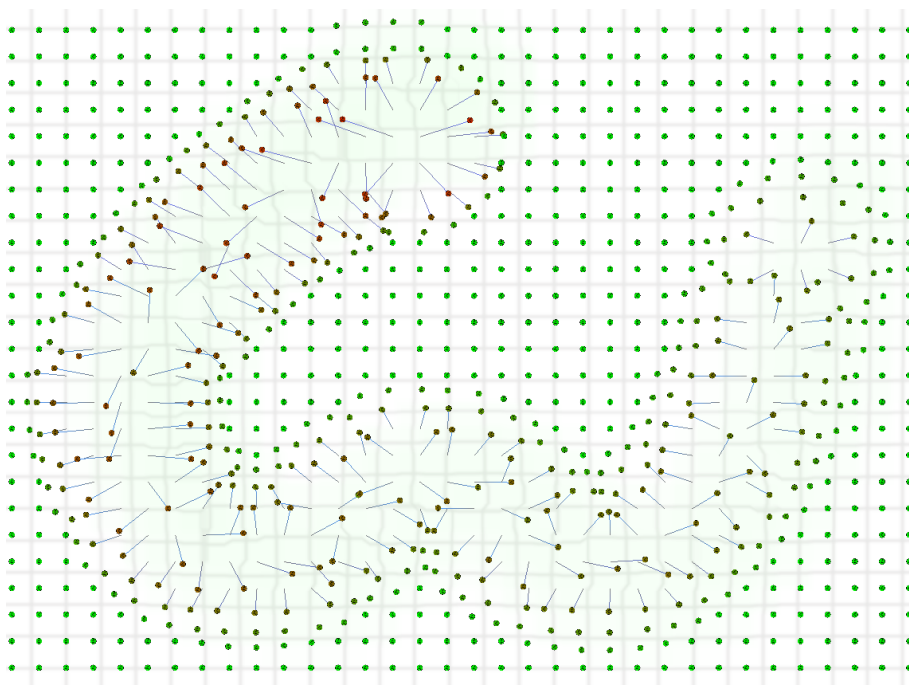


Figure 6.17: Illustration of a distortion grid with low attenuation.

7 Evaluation

The previous chapters introduced and motivated AudioPhield and discussed its design and implementation. In these chapters, most of the decisions made during the development of the system were well grounded on knowledge from various branches of research. This chapter will show if this knowledge was applied correctly – and thus, if the decisions were sound – by describing and analyzing a user study we conducted. It will also verify if we achieved the goals defined in chapter 4.

The chapter is thereby organized in the following way: First, we will explain which questions this user study is supposed to answer, and which answers we expect. Then we will describe the general setup and conduction of the user study. Information about the participants is herein included. The section afterwards covers the tasks, the users are asked to carry out. Then, we will present and discuss the results in the sections “Early findings”, “Task completion” and “Observations”. Finally, it will be discussed if, and how, the study answered the questions we asked in the beginning.

7.1 Aim and Expectations

AudioPhield was developed with a clear goal: A computer system was to be developed, which supports casual, explorative, and collaborative browsing of private music collections on a multi-touch tabletop display.

With the knowledge about (social) music consumption in chapter 3, we can refine and operationalize the task of casual and collaborative browsing: The system is supposed to support music talk as introduced in section 3.2.3. So, the first question the user study should answer is:

- *Does AudioPhield support music talk?*

Although this question is quite central, it might be hard to answer because the occurrence of music talk also depends heavily on outside factors like the relationships inside the user group (compare section 3.2). However, due to the fact that users are standing around a table that is dedicated solely to music playback at this time and due to probable rediscoveries of almost forgotten songs, which are bound to provoke music talk by conjuring up memories, we expect that AudioPhield will at least not hinder music talk.

Section 4 also revealed the core issues in the development. Whether these issues were addressed properly and solved by AudioPhield’s design, are further questions, which the user study shall answer. These are in detail:

Issue “Relate Pieces of Music”:

- *Does the chosen visualization of perceived song similarity as spatial proximity support casual browsing?*

Issue “Visualize Music Libraries”:

- *Does the depiction of a whole music library at once provide reasonable overview?*
- *Do focus areas in the form of fisheye lenses enable accessing details while preserving context?*
- *Is the information depiction reasonable, i.e., is the most relevant information for browsing present and readable?*

Issue “Enable Simultaneous, Collaborative Interaction on a Tabletop Display”:

- *Are users able to understand and use the integrated interaction possibilities?*
- *Can users interact simultaneously? Do they interfere with each other?*
- *Are the interaction techniques suitable for a tabletop display? Does the system exploit the table’s multi-touch capabilities?*

Since the design was essentially created to meet the requirements implicated in these questions, we expect positive answers to all of them.

While the previous questions were only concerned with the design of AudioPhield, i.e. with the system’s features as they should be, the user study is also supposed to verify if the implementation realizes this design sufficiently. So, a further open question is:

- *Is the software a sufficient realization of the developed design?*

For the most part, we think that the implementation will measure up to the design in the user study. One reason for that is, of course, that the design phase and implementation phase were not really isolated but interwoven enough to influence each other – even the evaluation phase caused minor adaptations (see below). The only major aspect where we believe that the software will fall short of the design, is the similarity computation because the data delivered by the extractors seems not detailed enough.

Due to the fact that the user study is performed using the implemented system as presented in 6, this question is not isolated from the questions above regarding the design. So, wherever AudioPhield fails to meet our expectations, we need to ask: Is it a flaw in the design or in the implementation?

7.2 General Study Design

We decided to conduct a rather informal, explorative study to answer the question catalog above. This form of study was chosen because of two reasons: Firstly, AudioPhield’s aim is to support casual activities instead of tasks with clear outcome like database queries; therefore, it is hard to operationalize to what degree the system is effective. Secondly, formal studies tend to force probands into specific behavior. We, on the other hand, were more interested in how users would interact with the system and with each other on their own. The concept of music talk, one of the key-behaviors to support, in itself requires that there is no special task to perform.

The general setup for the study was as follows: Probands were to interact with the tabletop display depicted in 6.1 on page 49. The device was therefore erected in a quiet room against a wall so that it is easily accessible from three sides. Albeit these surroundings can not be regarded “private” as postulated in the target “Usage Scenario” on page 19, they were at least convenient; also, the fact that the evaluation took place in the evening hours provided for a feeling of leisure time. Two speakers were set on top of the table and connected to the computer running AudioPhield. A camera was installed next to the table in such a way that it could oversee the movements of the users and the displayed contents at the same time. The camera surveillance seemed advised to allow us afterwards to review the activities of the probands with respect to selected aspects. Furthermore, the camera freed us from the necessity to watch over the probands’ shoulder all the time, and should have provided them with a stronger sense of privacy. Nevertheless, an evaluator was during the whole study in the same room to give instructions and make further observations. Figure 7.1 illustrates the experimental setup.

Six participants, five male one female, of various occupations were recruited from our circle of acquaintances. They are between 21 and 31 years of age. While their technical



Figure 7.1: Photo of the experimental setup.

experience reaches from rather low to very high, they estimated their experience with tabletop interfaces not beyond average. All of the probands have private libraries of digital music with sizes from 1500 to 10,000 songs. They access their collections frequently and play on average between 40 and 300 songs per week. Thus, the participants can all be considered to meet the requirement of being interested in music (see section 4.1). All but one of them were familiar with automated music recommendation- and selection-systems like “pandora”([@17]) or “musicIP”([@18]); three probands used them frequently, and one even delegated the task of music selection mainly to such automated systems.

To enable collaborative and social observations, they were divided into three pairs. In the following, we will consistently refer to these teams as A, B and C, and to their members as A1, A2, B1, and so on. All probands knew their team partner before the user study, but to varying degrees: Only the members of team C are close friends, the relations in the other pairs were effectively limited to encounters at various festivities. If this affiliation is close enough that the probands feel comfortable to present their private music library to each other – as the target usage scenario in section 4.1 assumes – will be one of the study’s results. Table 7.1 presents the participants in more detail.

Obviously, some parts of the evaluation can only be conducted with pairs of users. Other questions, however, can only be answered when one user interacts with the system alone. Still other questions might not be answered by the footage of the study at all. Therefore, we split the user study into three parts: First, a single-user episode shall give information about how usable the system is, and how sensible the depiction and placement of the icons are. Then in the second part, both team-members operate AudioPhield together; here, the evaluation is concerned with evolving social protocols, interferences and general communication. To ensure that both are already familiar with the interface at the beginning of this collaborative phase, both single-evaluations are executed before. The first user was therefore asked to wait outside during the second participant’s solitary test. Finally, the probands have to fill in a short questionnaire about their experiences.

The usage scenario, for which AudioPhield was developed, expects that users are, at least in part, familiar with the depicted music library. Therefore, all probands were asked to select 100–120 songs from their collection and hand them over to us a day before the study. They were supposed to follow the following criteria for this selection: The selection should preferably represent the proband’s taste in music; they were allowed to choose randomly as long as they are able to identify at least most songs by their artist

	A1	A2	B1	B2	C1	C2
Age	28	30	21	31	27	26
Gender	♂	♂	♀	♂	♂	♂
Technical Experience	++	++	–	+	+	++
Experience with tabletop interfaces	–	o	– –	o	– –	o
Size of private collection	7200	3000	1500	10000	7500	4000
Average songs played per week	300	100	80	40	60	100
Auto-selected songs (%)	0	10	0	60	0	30
Relation	loose		loose		close	

Table 7.1: Characteristics of the probands. Experiences are depicted on a scale with 5 steps from “no experience” (– –) to “somewhat experienced” (o) to “very experienced” (++).

and title. The selection may, but is not required to include favorite songs. The demand that the participants deliver their selection at least one day before the evaluation has two reasons. First of all, the automatic extraction of features takes up to ten seconds per song. While these 20 minutes could be annoying for the proband to wait before the study would start, the real time need arises with second reason: Since we were not convinced that the automatically extracted data would suffice for a good similarity-based layout, we refined the layout manually by moving the already placed songs (via a special interface, which was extra integrated for this purpose). If this measure was really necessary – and if the manual data was really better than the automatic – is also part of the study.

Unfortunately, the preselected music from the participants did not include all the information that AudioPhield was developed to display. Especially, it did not contain any data of how recently a song was played, how often it was played at all, and how it was rated³⁸. This means that two of the visualizations we had devised in the design phase were now obsolete, and, thus, their relevance and intelligibility had to remain unverified by this study.

Before the user study was conducted with the participants introduces above, we executed a sanity check with two associates from our university. As a result, minor bugs in the implementation could be found and fixed. Furthermore, obviously missing features were found and integrated; the grid in the background to visualize the fisheye’s effect, the possibility to jump inside a playing song and the coupling between the size of a focus area and its zoom strength are the most important of these. Of course, also the user study itself was revised to the final form presented in the next section.

7.3 Tasks

This section will discuss in detail which tasks the participants were asked to fulfill, and what answers were expected to be revealed by them.

³⁸Actually, some songs did contain rating data. However, mostly only few songs were tagged this way and – because the ratings were created by different programs – their encoding was not consistent.

7.3.1 Single User Tasks

The single user evaluation consists of six separate steps. Four of these steps are concerned with the two available interaction techniques ZoomFrames and SoapSpots. To minimize learn-effects, the succession of these tests alternates. So, three of the six participants evaluate the ZoomFrames first and then the SoapSpots, while for the other three the order is reversed.

In the first part of the solitary evaluation, we want to find out how “intuitive” the design is. This includes the question, to what extent AudioPhield reacts according to the user’s expectations, and which features and functions the proband is able to find on her own, i.e., the “discoverability” of the interface. Therefore, we start with a so called “zero-instruction-run”: When the subject enters the room, AudioPhield is already running and depicts her music collection with the manually enhanced layout. She is given no instructions but “please interact with the application and think aloud”; also, the evaluator is not allowed to provide her with any support. While the proband is interacting with AudioPhield, the evaluator takes notes, which features of AudioPhield were understood and which functions were found. He was provided for this task with a detailed task schedule, which also enlisted all discoverable items. This schedule is also included in this thesis as Appendix B. Since this is the proband’s first contact with the application, there are possible findings in three categories:

1. The starfield-like view (e.g., icons represent pieces of music)
2. Playback-interaction (e.g., tapping to start and stop the playback)
3. ZoomFrame- / SoapSpot-interaction (e.g., creation or movement of focus areas)

This phase continues until the subject is convinced that she has discovered all functions she is able to find. It may be aborted by the evaluator, too, when there is nothing left to discover or when the maximal time of ten minutes is reached.

In the next step, the researcher explains and demonstrates all functions of AudioPhield to the proband and verifies that the discovered items were really understood. All features not discovered or understood before are now tested for their learnability. Therefore, the evaluator sets the subject tasks which are only accomplishable when the latter uses the features to test; e.g., when the proband did not discover the playback via double-tap, she is assigned the task “please play at least five random songs”. The concrete list of tasks was previously defined in the task schedule (see Appendix B).

The following two steps essentially repeat the two stages delineated above with the not yet presented interaction technique to handle focus areas. So, subjects who interacted previously with the ZoomFrame view are now presented the SoapSpots, and vice versa. Also the task principle stays the same: A zero-instruction-run to assess the technique’s discoverability is followed by a series of concrete tasks to estimate its learnability.

The following test is a preference test: The subjects are given the possibility to adjust various aspects of the interface to their liking. Therefore, a wireless keyboard is brought to the table. By pressing different function keys, various variables are modified. The evaluator explains and demonstrates the effect of these variables; the system shows additionally a help-list of the functions assigned to the buttons. There are four ways to modify the interface:

1. Switch between ZoomFrame- and SoapSpots-interaction
2. Modify the plateau-size in three steps: No plateau, normal size, triple size.
3. Modify the distortion strength in three steps from weak to strong.

4. Turn the display of the origin-lines (the lines connecting icons in distortion areas to their original position) on or off.

The first option is just supposed to reveal which kind of interaction the probands enjoyed more. The second option shall uncover if our assumptions about the modified fisheye-functions (see section 6.3.4) are accurate. If not, the subject is expected to turn the plateau off. Is the third possible value chosen, then the user effectively was not content with the fisheye distortion since this setting modifies the focus areas to use primarily linear distortion. The modifications in the third variable are supposed to tell us if the user wishes to have more detail information (large distortion value) or rather more context information (small value). The last option will show if the users value the origin-lines as aids for understanding and interacting with the focus areas, or perceive them rather as clutter with limited use.

The last stage in the single-user phase is a data-influence-test. With this, we want to verify that our assumptions about the quality of the automated feature extraction were right, and thus the manual refinement necessary. Therefore, the participants are asked to examine the current (revised) layout again for a short time. Then, the layout is switched to the original, unmodified version. Now, without commenting on what was changed and how the different layouts were created, the researcher requests the subjects to examine the system again and tell their impressions. The test ends, when they have decided which layout they favor and why.

7.3.2 Pair Tasks

This part of the evaluation is concerned with collaborative and social aspects of the interaction with AudioPhield. The setup is similar to before with the difference that now the music selection of both team members is depicted on the interface. During this whole stage, the participants are allowed to modify the interface to their liking by the means presented in the preference test above. The stage is split roughly into two separate sections.

In the first half, we are interested in how the teams behave on their own. Therefore, the pair study starts with another zero-instruction-run. During this phase, we hope to gain insights about collaborative questions like “is the communication between the team members symmetrical or are there asymmetrical roles?” or especially “is there music talk?”. Also, we want to observe if social protocols evolve, and what conflicts they rule and how. After this instruction-less stage, the evaluator assigns special tasks to the team, which are supposed to induce certain behavior. So, the first task is “please introduce your music collection to your partner”. This task should clearly foster asymmetrical conversation because it implies that one proband adopts the active role of the presenter while the other is forced to act as audience. Another task shall force the participants to compare their musical taste and might therefore induce “music talk”-like communication. The instruction is “play songs to your partner which are good for dancing/relaxing/driving/working out”; the exact purpose the songs should have is decided ad hoc by the researcher to match the interests of the participants. The next task has a similar aim but should require more communication: Each team member is asked to close gaps in the musical education of their partner by playing songs to them which they do not know but should – for whatever reason. The next assignment shall coerce the participants into focussing their interactions on the music selection of their partners. Therefore, they are asked to choose three songs they would also like to have in their library.

The second part shall reveal to what extent it is possible to interact simultaneously with AudioPhield. Therefore, both subjects are explicitly asked to perform the following tasks at the same time. The first task is the one with arguably the lowest conflict potential: The evaluator requires the participants to select and play five songs they would include in

a playlist for commuting or driving. If the musical taste of the probands is not very similar, they can concentrate for this task on different parts of the surface. This is also possible in the next assignment, albeit under more stressful conditions: “This is a race. To win, you need to be the first to play five songs whose title begins with a ‘S’.” Here, we expect more interferences than in the first case. The final task is designed to cause as much conflict as possible. It is again a race but this time the participants are required to play songs of artists that need to be contained in both selections³⁹.

7.3.3 Questionnaire

After the team tasks the participants are asked to fill in a short questionnaire. Apart from general questions about their technical experience, music collection and music listening habits, this questionnaire contains a part with seven statements, to which the subjects could specify their agreement by a Likert scale with five levels ranging from “strongly disagree” to “strongly agree”. The sentences to rate are concerned with general impressions gathered during the interaction (e.g., “I generally enjoyed interacting with AudioPhield”) but also include rather special topics (e.g., “I had problems to read the radial texts”). Also, the questionnaire offered one page with three general, open ended questions for the participant’s favorite features of AudioPhield, its biggest flaws and general improvement ideas. See Appendix C for the complete questionnaire.

7.4 Results

The following section presents the most interesting results of the user study. First, we will examine how the participants were able to complete the tasks and what conclusions we derive from this. Then, a section follows about interesting general observations, which were not explicitly in the scope of any task. The findings here build also the basis for the final section “discussion”, which attempts to answer the question list in paragraph 7.1.

Note for the cites of participants used in this chapter: The probands spoke German during the study; for the sake of readability, however, their remarks will be translated into English in the following without explicit marks.

7.4.1 Task Completion

This section will reproduce how well the subjects were able to fulfill the tasks above. Also, their trials and expectations are documented here.

Single-User Tasks

All subjects understood the connection between pieces of music and icons almost immediately. They also interpreted the presented layout as meaningful, and expected icons close to each other to have something in common, albeit the interest and the interpretation varied: For example, subject B2 suggested that clusters of icons might be “topical linked” but did not investigate further. His team-partner, B1, suspected an ordering based on publication year and language and attested the ordering to be “not that logical”. Another user (A1) expected intuitively that proximity expresses general similarity and was so consumed by exploring the results (“ahh, here is the electro corner!”) that he nearly forgot to look for other functionalities of the system. Subject C1 also expected close songs to be similar and understood the ordering as based on genre and rhythm but was surprised to find two songs

³⁹To ensure that there are indeed songs of this kind present, the researcher had – if necessary – previously added songs from one collection to the other with respect the particular subject’s taste.



Figure 7.2: A proband tries stacking ZoomFrames to achieve music playback.

from the same artist, which actually sound rather unlike to us, quite apart from each other.

The attempts to play back a song varied greatly and showed a lot of imagination – one proband, for example, hoped that stacking four ZoomFrames on top of each other could work (see Figure 7.2). All test persons, however, first tried to activate playback by a single tap and were rather surprised to not achieve the expected result. Two users (B2, C2) beginning the study with the ZoomFrame interface also considered triggering playback by placing the hairlines on top of a song. Furthermore, subjects C1 and A2 tried dragging the VolumeWidget onto songs or songs onto the widget for this purpose. This already demonstrates that all users had at least initially difficulties to play music. However, all subjects eventually discovered one of the two possibilities to trigger playback eventually. Only one participant, C1, was able to find and understand the complete playback-system on his own. All of the users who only discovered the tag-hold method wished for a way to keep a song playing without having to keep the icon pressed; probands who found only double-tapping did not explicitly require another playback-mode but appreciated it as useful when it was shown to them. Only two users (A1, C1) discovered the possibility to jump inside a song to a certain position. The VolumeWidget drew especially at the beginning of the test the attention of all probands but B1, and most of them guessed either immediately (A1, B2, C2) or after few trials its function correctly. Thus, all users – apart from B1, who showed no interest in the widget – were able to change the volume and move the widget.

As for the discoverability of the techniques to create and handle focus areas, the results of the ZoomFrames and SoapSpots differed widely: All probands beginning with the ZoomFrames understood immediately that they could manipulate the frame by dragging the handle-spots – only subject B2 even tried touching the frame elsewhere. Also, all users found and used the possibility to resize and reorient the ZoomFrame. However, the rotation was primarily used to avoid occluding the focus area with the arm while dragging it. Subjects C2 and B2 expected the ZoomFrame to incorporate more functionality, especially regarding the hairlines in the center. Furthermore, all probands uncovered the way to create and destroy frames. The way to create SoapSpots was also discovered rapidly by all users, who began the study with this interaction mode, and their function was cor-

rectly understood as providing a way to magnify icons. Also, all test persons expected the SoapSpots to be movable and consecutively found this feature, too. The possibility to combine two focus areas to one was found just as easily. However, none of the participants discovered the possibility to scale the focus area or change its magnification by rotating it.

In the following stage, which was supposed to give information about the learnability of the system, all previously undiscovered functions proved to be at least learnable: The test persons had no difficulties to repeat the interactions demonstrated to them by the evaluator and could effortlessly fulfill their tasks. Only the changing of a SoapSpots' magnification required a repetition of the explanation in one case (B1). Another observed difficulty regards the double-tapping: While the subjects generally understood what was to do, they often executed the taps in too close succession. Thus, the two interactions were interpreted as one by the system, and the taps had no effect.

In the second zero-instruction-run all users who had used the SoapSpots before, had no difficulties to find and use all features of the ZoomFrames. However, the hairlines in the middle of the frame seemed to suggest additional functionality (C1, A2). The probands who had already experience with the ZoomFrames were quickly able to create, move, rescale and combine SoapSpots. The functionality to modify the magnification strength of the focus area, however, was not found. Subject C2 at least discovered that the color of a SoapSpot changes when the spot is rotated – but the effect of this color change was not understood. Then, when this functionality was demonstrated for the following learnability stage, all probands understood the feature and were able to fulfill the tasks given to them without difficulties.

In the preference test, the SoapSpots were preferred by the majority; only two probands (B1, C1) chose the ZoomFrames instead. Participant B1 commented on her choice and stated that she felt a little more in control of the interface here. The plateau-size was set by all users to the medium setting; thereby, subject A1 would have liked another level between the medium and the high setting. The distortion strength was set by four participants to the medium value. The other two probands (B2, A1) preferred the weakest and the strongest distortion, respectively. In both cases, the used SoapSpot was not set to its original zoom level but adjusted to a higher or lower setting; this might have influenced this choice.

The data-influence test showed that the manually refined placement indeed was more enjoyed. Only one user (B2) preferred the original, unaltered version – interestingly out of the same reason the other participants preferred the manual layout: In the automatic layout, the icons were generally closer together and built rather tight clusters. So, the primary advantage of the the manual refinement was that the icons were more spread out – and not that it was semantically superior.

Multi-User Tasks

After the multi-user part of the study was started, the probands were first confronted with the color-coding of the icons, which indicates ownership. The members of all teams had no difficulties to understand this connection and to assign a color to its owner. The probands were again advised that they may adjust the interface to their wishes as in the preference test before. This offering was generally declined the teams; only one pair changed to the ZoomFrame interaction after especially reminding them of this possibility during the test.

The first task-less minutes were spent by team A rather idly browsing the combined collection. They compared their musical taste by inspecting the layout and could thus rather quickly discover that they have little in common (A1: “we overlap only with the Eurythmics”). Apart from that the team was symmetrically and casually browsing the combined collections without much communication and without even playing music. Both participants focused their browsing thereby on their own songs. This behavior was es-

entially mirrored by team B; they seemed even more cautious than team A and, after they had found about the color-coding, were clearly waiting for instructions. Team C, whose members are closer friends, used this phase primarily to explore the possibilities of the interface together. Thus, they playfully tested how many SoapSpots the application can handle at the same time, or they expressed their wish for a function to directly copy the songs they like to their iPod. During this beginning phase music was played by both participants, obviously as ambient sound, and without hesitating if the partner might not like the choice. Then, they discovered that the density of icons of different colors gives information about similarities and differences in their musical taste. Afterwards, for a few minutes they explored collaboratively where their selections differ. During this exploration their communication was mostly symmetrical and both probands interacted simultaneously. This was frequently interrupted if one of the users found something he wanted to show to his partner. Also, these findings were accompanied by typical examples of music talk (C1: “That song is cool”, C2: “Hey, this reminds me of [...]”, C1: “Eew, that’s Après-Ski music!”).

The behavior in all groups changed with the first task (“Please introduce your music selection to your partner”): The communication and interaction was now clearly asymmetrical with one person presenting and the other watching. The presenter usually accompanied her explanations with the playback of songs which she deemed typical for a facet of their taste. The contributions of the test person currently in the passive role was often limited to short remarks to indicate that she is paying attention. Differences could be observed in the approach the test-persons took when introducing their selection: While the majority simply identified clusters on the interface and described the music located there, two users (C2, B2) based their presentation on their estimations of their musical taste and tried to find fitting examples afterwards. Subject A1 confirmed at the beginning of this task that one’s musical taste is something deeply personal (as stated in section 3.2.1) with the remark “Present my selection? How embarrassing!”. Also, probands seemed concerned with choosing the “right” specimen of an ingredient of their selection. Especially the presentations of team A were frequently interrupted by phases where the presenter seemed just to browse casually while both team members were obviously enjoying the currently playing music. At the end they seemed to have completely forgotten their task and browsed simultaneously again. Now, they were also beginning to engage in music talk. Their browsing behavior changed for a minute to using the *same* SoapSpot and moving it in turns.

The next assignment (“play songs to your partner which are good for relaxing”) was fulfilled by all teams rather fast and clearly asymmetrical; in the case of team B, subject B2 even stepped back from the table while B1 was choosing songs. Also, the probands played only as many songs as they were asked to. The following task to close “educational gaps” was conducted generally similar, albeit with more communication. This communication, however, was essentially limited to the question if the partner was familiar with a certain song and a short answer. Seemingly, only team B utilized the visualization and looked especially in areas of clear domination of one color for the questioned “gaps”. The following task (“find songs in your partner’s collection that you would also like to have”) proved essentially already fulfilled for the teams A and B: These participants had previously found songs of this kind and now had basically to rediscover them. The search for these songs were conducted simultaneously but isolated in the case of team B. The members of team A, on the other hand, mostly took turns in interacting but looked collaboratively for the same song. Only team C executed this task independently of previous findings. Interestingly, these participants did not browse simultaneously; instead, they took turns in selecting songs from their own collection to recommend them to their partner.

Since all teams had previously shown that simultaneous interaction on the interface is possible, the task to select songs for playlists for commuting or driving, was omitted during

the conduction. So, the probands were then assigned the task to find and play songs, whose title starts with a 'S', as fast as they could. Here, the teams showed significantly different characteristics. While in the teams B and C interferences, as e.g., the unintentional combining of a focus area with the focus area of the opponent, occurred, they were mostly accidents and the originator of the disturbance even apologized in some cases. Furthermore, the members of these teams kept their focus areas at their minimum size – seemingly to avoid collisions of this kind. Team A, on the other hand, abandoned this kind of politeness and magnified their SoapSpots to the size they deemed helpful for their browsing – regardless if this might disturb their opponent. Subject A2 even used his secondary hand to create and move SoapSpots at random in the area his partner was currently browsing just to disturb him – which caused understandably much complaining from his counterpart (“Hey, put that circle away!”). The second race essentially mirrored the first. Seemingly, the participants did not focus their search on the areas, where icons of both colors were to be found, but essentially relied on the knowledge from earlier tasks since all teams had previously identified common artists at one stage or another.

Questionnaire

The results of the first two parts of the questionnaire, which are concerned with demographics and music listening habits, can be found in Table 7.1. The ratings assigned to the statements in the next part, “Experiences with AudioPhield”, revealed that one of the central parts of AudioPhield’s design, the similarity depiction, found the approval of the participants: Five of them agreed to the sentence “The similarity depiction seemed sensible”, only subject C2 was undecided (rating 0). Similarly, the interface appeared to the participants not as overly cluttered (four times “somewhat disagree”), and they appreciated the overview of their partner’s taste it provided (three strong agreements). The probands were generally content with the interaction techniques: Most participants stated that they “somewhat agree” (value 1) to the sentence “I felt in control of the application”. A similar topic is touched by the statement “AudioPhield reacted sometimes unexpected to my inputs’. Here, only three participants “somewhat disagreed”, one felt undecided, and two even somewhat agreed. The radial texts were seemingly not readable enough: Four probands agreed to “I had problems with reading the radial texts”, one agreed even strongly. However, despite this flaw, all participants enjoyed interacting with AudioPhield generally. See Figure 7.3 for diagrams illustrating all these answers.

In the first open question in the last part of the questionnaire, users were asked to specify their favorite features of AudioPhield. Their answers are very similar and generally support our previous observations. Here, with four mentions, one of the most common items was the generally appealing look, phrased in different forms like “the fancy graphics” (A2) or “the general optical design” (B1). The general concept of a similarity-based visualization found also the approval of four participants. Three probands praised the easy and intuitive way to change the playback position inside the playing songs. Also three people counted the fisheye view among their favorite features. The multi-touch interaction was likewise mentioned three times. The VolumeWidget and the lines connecting songs of the same album were named by one proband each.

Among AudioPhield’s biggest flaws appeared the bad readability of the radial texts three times. Two users criticized that there is no function to jump directly to a known song. Another user saw as a flaw that the interface gets cluttered too fast. Finally, a way to create playlists was missed by yet another proband. The general improvement ideas essentially contain the elimination of the previously listed flaws. The only item not mentioned before is the wish for explicit labels on the field as a navigation aid.

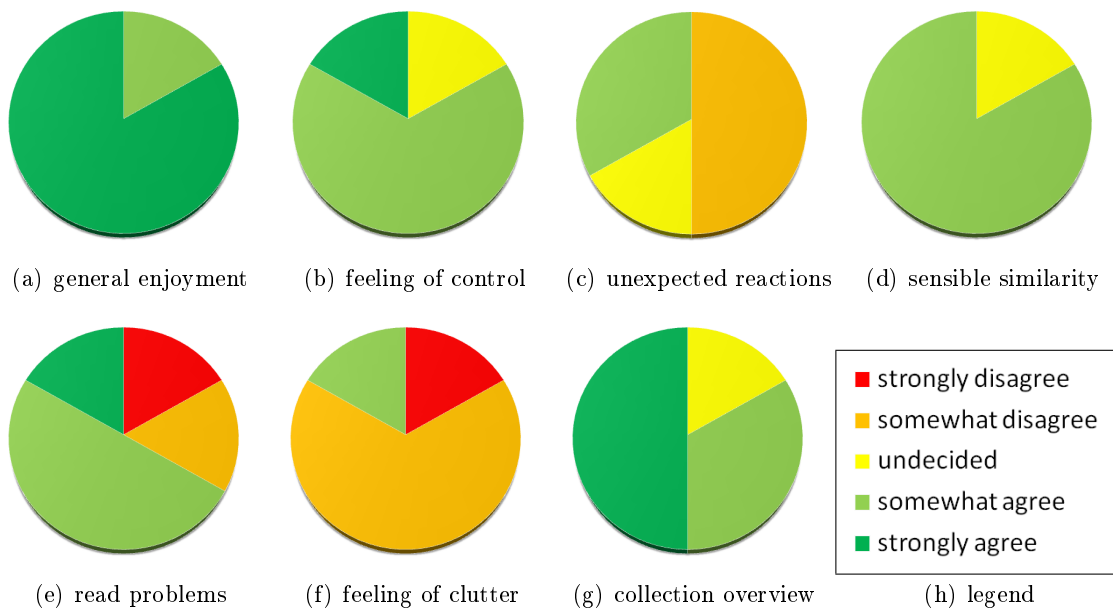


Figure 7.3: Diagrams of the ratings to “Experiences with AudioPhield”. The ordering mirrors the succession in the questionnaire (See Appendix C).

7.4.2 General Observations

This section enlists (in no particular order) all observations that can not be related to a specific task but are of interest nevertheless.

Standing positions: During the single-user phase all probands stood as expected in the center in front of the tabletop (as in Figure 7.2). In the multi-user setting, this was still the most taken place. The other participant usually stood at another side of the table facing the first user from the side. Standing side by side happened rarely, standing at opposite sides of the table was not once observed. The participants seemed sometimes reluctant to change their position and walk around the table – even if they had to reach out far for some interactions. This, however was strongly correlated with their current task: In situations where the users could expect to browse an area for a prolonged time, reaching out far was observed rarely.

Behavior dominated by closeness of relation: This was already to be expected from the knowledge presented in chapter 3.2.3. And indeed, team C, whose members were already close friends before the study, appeared to be more at ease when browsing and playing songs belonging to their partner’s selection. However, music talk could be observed in the communication of all teams. It seemed thereby correlated to how comfortable the probands felt in each other’s presence.

Learn effects: The ease of interaction of all probands showed generally significant improvement over time, especially regarding focus area handling. Unfortunately, the double-taps showed less improvement and were even at the end of the study frequently attempted unsuccessfully. Also, the participants learned the layout of the icons quickly and turned in most cases towards the right interface area when they were looking for a certain song or kind of music (e.g.: “I’m feeling like something more quiet now. That’s over there, right?” (B2)). In the multi-user phase, probands also sometimes taught each other ways

to interact better (e.g. “you need the make longer breaks between your taps” (B2)).

One-finger interaction: Although all users understood the capabilities of the tabletop to track an virtually limitless amount of arbitrarily shaped blobs, all probands interacted with the interface usually using only the index finger of their primary hand. Interactions requiring two input-points were mostly executed using both hands instead of two fingers of the same hand. When browsing areas of high icon density, the probands used sometimes two-fingered interaction to increase the input-precision, especially when using ZoomFrames.

Icon centering: In the beginning of the study, most probands tried to center an icon before interacting with it – an undertaking that is even with the plateau-enhanced fisheye challenging. Later, this behavior was rarely found; instead, they moved their focus areas just near enough to identify a song.

7.5 Identified Issues

The following is a list of all identified issues ordered roughly by their severity.

- **Missing classic search:** The need for a classic textual search arose in all observed multi-user phases: For example, subject B2 was reminded by a playing piece of music of another song, which possibly sounds very different and was therefore not located in the immediate proximity. Since free associating is one of the core attributes of music talk, this must be considered as a real shortcoming of the design.
- **Unreadable Radial Texts:** Apart from being pretty, the radial texts seemed to have no positive attribute. They caused arguably as much head-tilting as the usual straight lines of text, and were not nearly as readable. At the same time, these texts arguably use up more space than ordinary texts. Furthermore, the long-term observations of Ryall *et al.* in [70] suggest that “users ha[ve] no trouble comprehending small chunks of text that were improperly oriented”. One reason, why these texts were integrated, was to reduce the feeling that areas “belong” to a participant. However, issues of this kind did not seem to harm the communication in the user study in any way, and may thus be considered negligible.
- **Artist data is more important than song titles:** The probands virtually never identified songs – or interacted with them – when only the song title was displayed. Judging from their reactions, the title information in itself is not sufficient to identify the song. Furthermore, for fast navigation mostly the artist data was of concern. Thus, the artist should be displayed instead of the title around half-zoomed icons.
- **Similarity needs transparency:** Considering that there is no ground truth regarding the similarity of pieces of music, this issue does not come as a surprise. While the probands seemed to generally share the understanding of similarity with the evaluator, who refined the layout manually, there were still surprises and dissents. These could be eased and the interface at large could be better understandable if cues of any kind were given that could explain the reasons behind the layout.
- **Line trails:** Connecting a song with every other track of the same album led to more clutter and confusion than necessary. Since only two users selected whole albums for the study, this problem stayed rather hidden. An elegant solution might be if the songs of the same album were connected by a single “trail”, which could also indicate the playing order of the songs on the record.

- **Line misconceptions:** The lines connecting songs of the same album were frequently mistaken for origin-lines, and vice versa. This was in part caused by the poor color reproduction of the display, especially of narrow lines. But even if the color-coding had been better readable, this problem might have surfaced. Better distinguishability might be achieved by using curved or wider lines for the song connections.
- **Missing interaction-”catching”:** The simple call-sequence of `InputInterpreters` presented in chapter 6.2.2 contains no possibility to assign an input-ID to an interaction object. Thus, it happened sometimes that a user wanted to move her `ZoomFrame` and came near another `ZoomFrame` or the `VolumeWidget`. Then, she would suddenly drag one of the latter – much to her surprise. With little work, this problem should be easy to eliminate.
- **Missing additional information:** Currently, `AudioPhield` presents to a selected song only which other songs belong to the same album – *which* album that is, is not accessible. During the evaluation, the need for additional information like this appeared only sporadically; in a real-world environment, however, there is arguably more demand for additional data.
- **Too small SoapSpots:** The ring indicating the interaction-area of a `SoapSpot` was often misunderstood as visual cue for input. Thus, probands, who wanted, e.g., to resize the spot, tried to grab it on the ring – and created new focus areas instead. So, either the visualization of the `SoapSpots` needs to be changed to indicate that it may be touched anywhere or the interaction area should be enlarged to also contain the ring completely.
- **Unnecessary timeout:** `SoapSpots` vanish if they are not interacted with after a certain time. This (rather rare) event was in virtually all observed cases unwelcome and considered annoying. Since unwanted `SoapSpots` can easily be deleted by just combining them, this function should be removed.

7.6 Discussion

In this final section of the evaluation chapter, we will explicitly analyze if the questions asked in 7.1 were answered by the study, how these answers read and if they confirm our expectations.

The first question was *Does AudioPhield support music talk?*. Since a fair amount of music talk was observed, the answer here is that the system at least not frustrates it. A comparative study would be necessary to confirm that `AudioPhield` supports music talk better than classic systems – however, based on our own experiences, we feel optimistic that our system would outmatch classic music playback software, even if it was also running on a tabletop display. Nevertheless, a classic textual search should be somehow integrated to support music talk even better.

The next question was *Does the chosen visualization of perceived song similarity as spatial proximity support casual browsing?*. Here, the answer is a clear ‘yes’: Most of the browsing we observed appeared to be casual. Some probands (A1, B1) even explicitly remarked that interacting with `AudioPhield` was a great way to “durch die Sammlung schmökern” what might be best translated as “to browse casually”. The similarity depiction seemed especially appreciated if users were comparing their musical taste. Also, probands relied on the similarity-based layout when looking for music of a special kind, e.g., during the task “present your music collection” – as expected.

The next two questions, *Does the depiction of a whole music library at once provide reasonable overview?* and *Do focus areas in the form of fisheye lenses enable accessing details while preserving context?*, can be answered together. The participants' behavior during the evaluation showed clearly that they were able to navigate the interface without getting lost; at the same time, they were able to access more detailed information through focus areas and had seemingly no trouble relating this detail-information to its surroundings. Additional indicators for a sufficient overview display were remarks like "one can immediately see where we have the same musical taste" (A1) as well as the high agreement the following statement in the questionnaire received: "I got a goof overview of my partner's music taste". The depiction of detail-information may have suffered from the bad readability of the radial texts but was otherwise little criticized. The fisheye-effect at least, was mentioned several times as one of a subject's favorite features of AudioPhield.

The study delivered no clear answer to the question "Is the information depiction reasonable, i.e., is the most relevant information for browsing present and readable?". This is in part due to the fact that important information, like play-counts and recency of last playback, could not be tested in this study since the information was simply not available. The general predominant disagreement with the statement "The interface felt cluttered and confusing" suggests that most probands were rather happy with the information visualization. One subject wished for additional information to selected songs. Apart from that, the most relevant information seemed to be present, albeit its depiction could be improved, e.g., by replacing the radial texts and displaying artist- before title-information.

Regarding "Are users able to understand and use the integrated interaction possibilities?", many indicators exist that this can be answered with a 'yes'. Many of the features could be discovered by the probands on their own, and the undiscovered features were easily learned – albeit they were rarely used in some cases. For example, only two users (A2, C2) made use of the possibility to modify the zoom-level of SoapSpots. Also, the double-tapping should be revised to make it easier to trigger. Apart from these minor issues, the interaction offered little reason for criticism. The results from the questionnaire regarding this topic backs this statement up: The participants felt in control of the application most of the time.

The next question "Can users interact simultaneously? Do they interfere with each other?" can be answered similar positively. Simultaneous interaction occurred frequently, interferences rarely by contrast. The fact that the user disturbed each other barely stems to a good part of the fact that social protocols emerged in situations of conflict: E.g., when two users wanted to explore the same area of the interface, they sometimes shared one focus area.

Considering the result from the previous questions, it is no surprise that we feel comfortable to answer the query "Are the interaction techniques suitable for a tabletop display? Does the system exploit the table's multi-touch capabilities?" again positively. The observed high discoverability suggests that most interaction techniques seemed to have been expected in a similar form, and thus suited a tabletop device. Considering that the participants had little to no experience with tabletop interfaces, this statement deserves even more recognition. AudioPhield does not exploit all possibilities of the used hardware – our own design ideas prove that: Interaction techniques need not to be limited to small input spots that are interpreted as points; approaches like the CookieDough interface show how the tabletop could also interpret arbitrarily shaped interaction areas. On the other hand, the participants even refrained largely from using more than their forefingers for input – what again might stem from their intensive experiences with desktop PCs and little experience with tabletop devices. Further studies will have to show if future interfaces will still be operated just by fingertips.

Regarding the last question, "Is the software a sufficient realization of the developed

design”, we found little indicators that would suggest otherwise. Virtually all issues – apart from the missing “input catching” – were design related. Also, the praise of the general look of AudioPhield from participants is to a certain extent a praise of the implementation.

How valid are these findings? As explained in the “General Study Design” (page 76) the conducted user study could not exactly reproduce the usage scenario for AudioPhield – in part because of the simple fact that it is a user study, and the probands did not meet in private surroundings for leisure activities (although they did seem to enjoy themselves). Unobtrusive observations in real private surroundings should be more conclusive. Furthermore, not all features of AudioPhield could be evaluated. The here missing depiction of playback recencies and play-counts might further support music talk and casual browsing – after all, this information relates pieces of music in a completely different aspect. On the other hand, of course, the chosen visualizations for this data might also harm the general readability of the interface. Nevertheless, the user study delivered valuable insights and the participants’ relaxed behavior suggests that most of the findings above would appear in more fitting surroundings, too.

Thus, to sum up this chapter, with AudioPhield we may not have achieved our goals utterly but came at least close: AudioPhield indeed supports casual browsing of private music collections on a tabletop display, even if there are still improvement potentials.

8 Conclusion

To round off this thesis, this chapter will briefly summarize the previous explanations and indicate possible directions for future work.

8.1 Summarization

This thesis presented AudioPhield, a system for collaborative casual browsing of music collections on tabletop displays. In the introduction, we showed that listening to music is a deeply emotional and inherently social process – which is not supported by usual software for music playback. Starting from this initial motivation, we esteemed it plausible that this condition correlates with the fact that usual software to handle music libraries is tailored to desktop PCs, which in itself are inapt for simultaneous collaborative tasks. Consequently, another computing principle was introduced, the direct-touch tabletop systems. These seem better suited for our purpose. This motivated AudioPhield’s aim to run on such a device.

The following chapter presented and discussed the work done by other researchers in three disciplines, which were identified as the core topics for the development of AudioPhield: Information visualization, similarity-based music browsing and social browsing on tabletop devices.

Chapter 3 examined how people usually consume music. It started by analyzing typical organizations of music collections. Then, the typically rather random, undirected strategies of accessing private music libraries were discussed. Further, it was shown that individuals as well as groups use music to define their identity. Then, the typical behavior of acquaintances regarding choosing and listening to music was analyzed. This built also the basis for the introduction of the term “music talk”, which describes an unstructured process of casually discussing topics related to the playing music. The chapter was closed with the topic of sharing music among friends. Here, an asymmetrical form of music talk was diagnosed in the usual communication manners surrounding this task.

With the knowledge accumulated in the previous chapters, the targeted usage scenario as well as the main issues for the design and development were defined in chapter 4: AudioPhield should be used in private surroundings by people who know each other. As central challenges were identified the relating of songs, the visualization of music libraries and enabling simultaneous interaction on a tabletop display.

Chapter 5 then illustrates the process of designing an interface that is supposed to achieve the just defined goals. We deducted from the music consumption chapter that the perceived similarity between pieces of music is one of the most important features for casual browsing. Therefore, it was decided that these relations should build the core of the visualizations in AudioPhield. The data necessary for this purpose should be acquired through algorithms that automatically estimate meaningful features of each song. Because spatial proximity was revealed to be the most significant way to visualize information, songs should be depicted as icons in such a way that the perceived similarity determines the distance between the icons. After discussing different approaches, it emerged that self-organizing maps are well suited for the task of building a similarity-based layout. As next step, the visualization of music libraries was discussed. Since the display was required to deliver overview information and detail information, graphical fisheye lenses became part of the design. It followed a determination of what should be encoded in the design of the icons. To show the necessary information to identify a piece of music – the name of the artist and the title –, we devised radial texts to avoid the orientation-issue of tabletop interfaces. Furthermore, the color of an icon should indicate the music library to which the represented song belongs, opacity should encode the recency of a the last

playback of a piece of music, and rotation speed should correlate with its rating. The rest of the chapter was concerned with interaction techniques. Three different systems to create and manipulate fisheye lenses were presented: The rather classic ZoomFrames with two handle-spots, the SoapSpots that allowed to create new focus areas by a single tap, and the unimplemented CookieDough, which essentially simulates a directly manipulatable viscous dough. Afterwards, ways to play and pause music were defined. There are two possibilities. Firstly, an icon may be tapped twice to start usual playback. Secondly, the user can keep an icon pressed for a short time; then the playback would start and continue for as long as the pressing is not released. This was followed by a description of how in-song position seeking is realized: After the tap-hold interaction moving the interacting finger around the icon sets the playback position. The introduction of the VolumeWidget to control playback volume concluded the chapter.

The following chapter “Implementation” started with a description of the used hardware. The section afterwards presented the general software design in terms of used technologies and architecture. Then selected implementation aspects were discussed. These were the usage of folksonomy data instead of content-based algorithms, the peculiarities of the SOM, a force-based algorithm to improve the layout, and the indirect way to realize fisheye lenses by utilizing a distortion grid.

Finally, in chapter 7 we presented an explorative user study conducted to verify the previous design decisions. The study was split into three parts. First, probands were interacting solitary with the interface. At the beginning of this phase, they were not given any instructions to test the discoverability of the interface. Later tasks were tailored to certain aspects of AudioPhield. Then, the participants interacted with the interface as pairs. After another instruction-less period, they were asked to perform tasks forcing them to interact with each other and with the system in certain ways, e.g., to test how well simultaneous interactions are possible. Finally, probands were asked to fill in questionnaires. The following analysis of the results revealed a few design issues, but showed also that with AudioPhield the goal to create a system to support casual collaborative browsing was essentially accomplished.

8.2 Future Work

There are some obvious ways how AudioPhield could be developed further. Removing the issues enlisted in chapter 7.5, for example, would be a good start. Also, better ways to collect meaningful data for songs to achieve better placements would be another. While here many interesting matters are hidden (e.g., providing AudioPhield with rather classic searches might be a poster use-case for query-by-humming systems), we nevertheless want to use these final lines to hint the untapped potential of the ideas behind AudioPhield.

The usage scenario for this thesis was intentionally chosen quite narrow. Otherwise, the implementation would have come never to an end, and we would still be discussing ideas. One should not assume that the concept is limited to this scenario, though. A similar table as the one used here may, for example, be erected inside a record store. Then, customers could simply feed their favorite song to the system and see immediately what similar songs are available. This could even be enhanced with data from other music enthusiasts all around the world: Instead of playback recency, the system could visualize, e.g., how often the song was purchased in the last hours. Perhaps it would be even more interesting for customers if they could connect their iPod directly to the system and purchase songs by a simple dragging gesture. A system similar to AudioPhield might also be of high value for single-user environments: Consider yourself coming home from a day that has been a little too long, and you desire the right tunes to help you relax. You *could* scroll through your lists of albums until you find something that fits; or you could just draw a circle on

the screen around the area that fits your mood. Of course, these scenarios make rather different demands on the system than the one we considered here – but the general concept of AudioPhield would in our opinion be up to it.

CD content

`\AudioPhield`

Contains the final version of AudioPhield as source code.

`\footage`

Parts of the video material recorded during the user study

`\references`

Contains all freely available papers and webpages of the ‘‘references’’ list

`\thesis`

`\PDF`

Contains the thesis as .pdf

`\LaTeX`

Contains the LaTeX files of the thesis.

Note: README.txt explains how the PDF can be created from the LaTeX files.

List of Abbreviations

API	Application Programming Interface
bpm	beats per minute
DOI	Degree of interest
e.g.	Latin: <i>exempli gratia</i> : for example
et al.	Latin: <i>et alii</i> : and others
f.	and the following page
ff.	and the following pages
FTIR	Frustrated Total Internal Reflection
GUI	Graphical User Interface
i.e.	Latin: <i>id est</i> : that is
InfoViz	Information Visualization
IoM	Islands of Music
KISS	Acronym: Keep it simple, stupid
MIR	Music Information Retrieval
p.	page
PCM	Pulse-Code-Modulation
SOM	Self-Organizing Map
STFT	Short Time Fourier Transformation
XNA	XNA's Not Acronymed

List of Figures

1.1	AudioPhield screenshot	3
2.1	FilmFinder screenshot	6
2.2	Fisheye-transformed photo.	8
5.1	Effectivity-ranking for display dimensions	23
5.2	Comparison between the visualizations Starfield and SOM	27
5.3	Screenshot of a prototype with a central fisheye lens.	29
5.4	Examples of fisheye distortions	30
5.5	Illustration of an interface with fisheye views	31
5.6	Illustration of an interface with visually emphasized fisheye views	32
5.7	A field with two music libraries	34
5.8	Examples of icon-visualizations	36
5.9	Movement of a ZoomFrame.	38
5.10	Size-zoom coupling	39
5.11	Rotation and resizing of a ZoomFrame.	39
5.12	Creation and deletion of a ZoomFrame	40
5.13	Creation and movement of a SoapSpot.	41
5.14	Rotation and resizing of a SoapSpot.	41
5.15	Combination of two SoapSpots.	42
5.16	Illustration of the CookieDough interface.	43
5.17	Simple playback	44
5.18	Scan-Playback	46
5.19	In-song seeking	46
5.20	Volume manipulation.	47
6.1	Frustrated Total Internal Reflection	49
6.2	Measures and photo of the used tabletop device.	50
6.3	Image processing in Touchlib.	52
6.4	AudioPhield’s architecture: Class-diagram.	53
6.5	Screenshot of the DataHandler.	55
6.6	Illustration of the encoding robustness of selected features.	58
6.7	Screenshot of the music annotation tool.	59
6.8	Two exemplary beat histograms.	61
6.9	Decision tree of tag extractors.	64
6.10	Illustration of the SOM learning process	66
6.11	Screenshot of the SOMpregantor.	68
6.12	Illustration of the forces exerted on two items.	68
6.13	Illustration of the effect of the spring-algorithm.	69
6.14	Transformation and magnification function of the default fisheye view.	70
6.15	Transformation and magnification function of fisheye view with plateau.	71
6.16	Illustration of low-resolution distortion grid.	72
6.17	Illustration of a distortion grid with low attenuation.	73
7.1	Photo of the experimental setup.	77
7.2	A proband tries stacking ZoomFrames to achieve music playback.	82
7.3	Diagrams of the ratings to “Experiences with AudioPhield”.	86

List of Tables

3.1	Typical organization of private music collections	14
5.1	Characteristics of various InfoViz channels	24
5.2	Separability of some display dimension pairs.	25
7.1	Characteristics of the probands.	78

References

- [1] Christopher Ahlberg and Ben Shneiderman. Visual information seeking: tight coupling of dynamic query filters with starfield displays. In *CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 313–317, New York, NY, USA, 1994. ACM.
- [2] Christopher Ahlberg, Ben Shneiderman, and Christopher Williamson. Dynamic queries: database searching by direct manipulation. In *CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 669–670, New York, NY, USA, 1992. ACM.
- [3] Jean-Julien Aucouturier and Francois Pachet. Improving Timbre Similarity : How high's the sky? *Journal of Negative Results in Speech and Audio Sciences*, 1(1), 2004.
- [4] Arianna Bassoli, Julian Moore, and Stefan Agamanolis. tuna: Socialising music sharing on the move. In Kenton O'Hara and Barry Brown, editors, *Consuming Music Together: Social and Collaborative Aspects of Music Consumption Technologies*, volume 35 of *Computer Supported Cooperative Work*, chapter 8, pages 151–172. Springer, Dordrecht, The Netherlands, 2006.
- [5] Hrvoje Benko, Andrew Wilson, and Patrick Baudisch. Precise selection techniques for multi-touch screens. In *CHI*, pages 1263–1272, Montréal, Québec, Canada, April 2006.
- [6] Ingwer Borg and Patrick Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer, 2005.
- [7] Barry Brown and Abigail Sellen. Sharing and listening to music. In Kenton O'Hara and Barry Brown, editors, *Consuming Music Together: Social and Collaborative Aspects of Music Consumption Technologies*, volume 35 of *Computer Supported Cooperative Work*, chapter 3, pages 37–56. Springer, Dordrecht, The Netherlands, 2006.
- [8] Thorsten Buering, Jens Gerken, and Harald Reiterer. User interaction with scatterplots on small screens - a comparative evaluation of geometric-semantic zoom and fisheye distortion. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):829–836, 2006.
- [9] Michael Bull. Investigating the culture of mobile listening: From walkman to ipod. In Kenton O'Hara and Barry Brown, editors, *Consuming Music Together: Social and Collaborative Aspects of Music Consumption Technologies*, volume 35 of *Computer Supported Cooperative Work*, chapter 7, pages 131–149. Springer, Dordrecht, The Netherlands, 2006.
- [10] William Buxton. A three-state model of graphical input. In *INTERACT '90: Proceedings of the IFIP TC13 Third International Conference on Human-Computer Interaction*, pages 449–456, Amsterdam, The Netherlands, The Netherlands, 1990. North-Holland Publishing Co.
- [11] Kim Campbell. The death of the album? *The Christian Science Monitor*, (14), November 2003.
- [12] Stuart Card, Jock Mackinlay, and Ben Shneiderman. *Using vision to think*, pages 579–581. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

- [13] Sheelagh Carpendale. *A framework for elastic presentation space*. PhD thesis, Simon Fraser University, Burnaby, BC, Canada, Canada, 1999.
- [14] Sheelagh Carpendale, David Cowperthwaite, and Frank Fracchia. Bringing the advantages of 3D distortion viewing into focus. In *Proc. IEEE Symp. Information Visualization, InfoVis*, pages 17–20, 1998.
- [15] Patrick Chiu, Jeffrey Huang, Maribeth Back, Nicholas Diakopoulos, John Doherty, Wolf Polak, and Xiaohua Sun. mTable: Browsing Photos and Videos on a Tabletop System. In *ACM International Conference on Multimedia '08*, October 2008.
- [16] William Cleveland and Robert McGill. Graphical perception: Theory, experimentation and application to the development of graphical methods. *Journal of the American Statistical Association*, (79):531–554, September 1984.
- [17] Andrew Crossen and Jay Budzik. Promoting social interaction in public spaces: The flytrap active environment. In Kenton O'Hara and Barry Brown, editors, *Consuming Music Together: Social and Collaborative Aspects of Music Consumption Technologies*, volume 35 of *Computer Supported Cooperative Work*, chapter 6, pages 111–128. Springer, Dordrecht, The Netherlands, 2006.
- [18] Sally Cunningham, David Bainbridge, and Dana McKay. Finding new music : a diary study of everyday encounters with novel songs. In *ISMIR '07, 8th International Conference on Music Information Retrieval*, pages 83–88, Vienna, Austria, September 2007.
- [19] Sally Cunningham, Stephen Downie, and David Bainbridge. "the pain, the pain": Modelling music information behavior and the songs we hate. In *ISMIR 2005, 6th International Conference on Music Information Retrieval*, pages 474–477, London, UK, September 2005.
- [20] Sally Cunningham, Steve Jones, and Matt Jones. Organizing digital music for use: an examination of personal music collections. In *ISMIR 2004, 5th International Conference on Music Information Retrieval*, pages 447–455, Barcelona, Spain, October 2004.
- [21] Scott Deerwester, Susan Dumais, George Furnas, Thomas Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.
- [22] Asaf Degani, Michael Shafto, and Alex Kirlik. Modes in human-machine systems: Review, classification, and application. *International Journal of Aviation Psychology*, 9(2):125–138, 1999.
- [23] Tia DeNora. Music and emotion in real time. In Kenton O'Hara and Barry Brown, editors, *Consuming Music Together: Social and Collaborative Aspects of Music Consumption Technologies*, volume 35 of *Computer Supported Cooperative Work*, chapter 2, pages 19–33. Springer, Dordrecht, The Netherlands, 2006.
- [24] Elena Deza and Michel-Marie Deza. *Dictionary of Distances*. Elsevier, 2006.
- [25] Paul Dietz and Darren Leigh. Diamondtouch: a multi-user touch technology. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 219–226, New York, NY, USA, 2001. ACM.

- [26] Stephen Downie. Music information retrieval. *Annual Review of Information Science and Technology* 37, pages 295–340, 2003.
- [27] Daniel Ellis, Brian Whitman, Adam Berenzweig, and Steve Lawrence. The quest for ground truth in musical artist similarity. In *Proceedings of 3rd International Symposium on Music Information Retrieval (ISMIR 2002)*, pages 170–177, Paris, France, October 2002.
- [28] Bernhard Feiten and Stefan Günzel. Automatic indexing of a sound database using self-organizing neural nets. *Computer Music Journal*, 18(3):53–65, Fall 1994.
- [29] Clifton Forlines and Chia Shen. Dtlens: multi-user tabletop spatial data exploration. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 119–122, New York, NY, USA, 2005. ACM.
- [30] David Frohlich, Allan Kuchinsky, Celine Pering, Abbe Don, and Steven Ariss. Requirements for photoware. In *CSCW '02: Proceedings of the 2002 ACM conference on Computer supported cooperative work*, pages 166–175, New York, NY, USA, 2002. ACM.
- [31] George Furnas. Generalized Fisheye Views. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI'86)*, pages 16–23, New York, 1986. ACM Press.
- [32] Jefferson Han. Low-cost multi-touch sensing through frustrated total internal reflection. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 115–118, New York, NY, USA, 2005. ACM Press.
- [33] Mark Hancock, Sheelagh Carpendale, Frederic Vernier, Daniel Wigdor, and Chia Shen. Rotation and translation mechanisms for tabletop interaction. In *TABLETOP '06: Proceedings of the First IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, pages 79–88, Washington, DC, USA, 2006. IEEE Computer Society.
- [34] Kate Hevner. Experimental studies of the elements of expression in music. *American Journal of Psychology*, 48:246–268, 1936.
- [35] Otmar Hilliges, Dominikus Baur, and Andreas Butz. Photohelix: Browsing, Sorting and Sharing Digital Photo Collections. In *To appear in Proceedings of the 2nd IEEE Tabletop Workshop, Newport, RI, USA*, October 2007.
- [36] Otmar Hilliges, Phillipp Holzer, Rene Klüber, and Andreas Butz. Audioradar: A metaphorical visualization for the navigation of large music collections. In *Proceedings of the International Symposium on Smart Graphics 2006, Vancouver Canada*, July 2006.
- [37] Uta Hinrichs, Sheelagh Carpendale, and Stacey Scott. Evaluating the effects of fluid interface components on tabletop collaboration. In *AVI '06: Proceedings of the working conference on Advanced visual interfaces*, pages 27–34, 2006.
- [38] Stephen Hitchner, Jennifer Murdoch, and George Tzanetakis. Music browsing using a tabletop display. In *Proceedings of ISMIR 2007*, pages 175–176, Vienna, Austria, September 2007.
- [39] Peter Hlavac. Innovative user interfaces for accessing music on mobile devices. Master's thesis, Vienna University of Technology, Austria, March 2007.

- [40] Lars Holmquist. Focus+context visualization with flip zooming and the zoom browser. In *CHI '97: CHI '97 extended abstracts on Human factors in computing systems*, pages 263–264, New York, NY, USA, 1997. ACM.
- [41] Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. Information Retrieval in Folksonomies: Search and Ranking. In *Proceedings of the 3rd European Semantic Web Conference*, LNCS, pages 411–426, Budva, Montenegro, June 2006. Springer.
- [42] Maria Håkansson, Mattias Jacobsson, and Lars Holmquist. Designing a mobile music sharing system based on emergent properties. In *Proceedings of the 2005 International Conference on Active Media Technology (AMT 2005)*, May 2005.
- [43] Samuel Kaski. *Data Exploration Using Self-Organizing Maps*. Acta polytechnica scandinavica ma 82, Neural Networks Research Center, Helsinki University of Technology, Espoo, Finland, 1997.
- [44] Peter Knees, Markus Schedl, Tim Pohle, and Gerhard Widmer. An innovative three-dimensional user interface for exploring music collections enriched. In *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia*, pages 17–24, New York, NY, USA, 2006. ACM.
- [45] Kurt Koffka. *Principles of Gestalt Psychology*. Lund Humphries, London, UK, 1935.
- [46] Teuvo Kohonen. Construction of similarity diagrams for phonemes by a self-organizing algorithm. Technical report, Helsinki University of Technology, Espoo, Finland, 1981. Report TKK-F-A463.
- [47] Teuvo Kohonen. *Self-Organizing Maps*. Springer, 3 edition, January 2001.
- [48] Myron Krueger, Thomas Gionfriddo, and Katrin Hinrichsen. VIDEOPLACE – an artificial reality. *SIGCHI Bulletin*, 16(4):35–40, 1985.
- [49] Russell Kruger, Sheelagh Carpendale, Stacey Scott, and Saul Greenberg. How people use orientation on tables: comprehension, coordination and communication. In *GROUP '03: Proceedings of the 2003 international ACM SIGGROUP conference on Supporting group work*, pages 369–378, New York, NY, USA, 2003. ACM.
- [50] Paul Lamere. Search inside the music: Using signal processing, machine learning, and 3d visualizations to discover new music. Slides at 2007 JavaOne Conference, 2007.
- [51] Paul Lamere and Douglas Eck. Using 3d visualizations to explore and discover music. Whitepaper, Sun Labs, Sun Microsystems, Inc., June 2007.
- [52] John Lamping, Ramana Rao, and Peter Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 401–408, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.
- [53] Butler Lampson. Hints for computer system design. In *SOSP '83: Proceedings of the ninth ACM symposium on Operating systems principles*, pages 33–48, New York, NY, USA, 1983. ACM.
- [54] Ying Leung and Mark Apperley. A review and taxonomy of distortion-oriented presentation techniques. *ACM Transactions on Computer-Human Interaction*, 1(2):126–160, 1994.

- [55] Jock Mackinlay. Automating the design of graphical presentations of relational information. *ACM Trans. Graph.*, 5(2):110–141, 1986.
- [56] Ali Mazalek, Matthew Reynolds, and Glorianna Davenport. Tviews: An extensible architecture for multiuser digital media tables. *IEEE Computer Graphics Applications*, 26(5):47–55, 2006.
- [57] Daniel McEnnis and Sally Cunningham. Sociology and music recommendation systems. In *Proceedings of 8th International Conference on Music Information Retrieval (ISMIR '07)*, Vienna, Austria, 2007.
- [58] Fabian Mörchen, Alfred Ultsch, Mario Nöcker, and Christian Stamm. Databionic Visualization of Music Collections According to Perceptual Distance. In *ISMIR 2005, 6th International Conference on Music Information Retrieval*, pages 396–403, London, UK, September 2005.
- [59] Kenton O'Hara and Barry Brown. Consuming music together: Introduction and overview. In Kenton O'Hara and Barry Brown, editors, *Consuming Music Together: Social and Collaborative Aspects of Music Consumption Technologies*, volume 35 of *Computer Supported Cooperative Work*, chapter 1, pages 3–17. Springer, Dordrecht, The Netherlands, 2006.
- [60] Kenton O'Hara, Matthew Lipson, Axel Unger, Huw Jeffries, Marcel Jansen, and Peter Macer. Distributing the process of music choice in public spaces. In Kenton O'Hara and Barry Brown, editors, *Consuming Music Together: Social and Collaborative Aspects of Music Consumption Technologies*, volume 35 of *Computer Supported Cooperative Work*, chapter 5, pages 87–109. Springer, Dordrecht, The Netherlands, 2006.
- [61] Nicola Orio. Music retrieval: A tutorial and review. *Foundations and Trends in Information Retrieval*, 1(1):1–90, 2006.
- [62] François Pachet, Amaury La Burthe, Aymeric Zils, and Jean-Julien Aucouturier. Popular music access: the sony music browser. *Journal of the American Society for Information Science*, 55(12):1037–1044, 2004.
- [63] Elias Pampalk. Islands of music: Analysis, organization, and visualization of music archives. Master's thesis, TU Wien, 2001.
- [64] Joseph Paradiso. Tracking contact and free gesture across large interactive surfaces. *Commun. ACM*, 46(7):62–69, 2003.
- [65] Karl Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh and Dublin Philosophical Magazine and Journal of Science*, 6(2):559–572, 1901.
- [66] Tim Pohle, Elias Pampalk, and Gerhard Widmer. Evaluation of frequently used audio features for classification of music into perceptual categories. In *Proceedings of the Fourth International Workshop on Content-Based Multimedia Indexing*, 2005.
- [67] Jun Rekimoto. Smartskin: an infrastructure for freehand manipulation on interactive surfaces. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 113–120, New York, NY, USA, 2002. ACM.

- [68] British Music Rights and the University of Hertfordshire. Music experience and behaviour in young people – main findings and conclusions. Technical report, University of Hertfordshire, June 2008.
- [69] Yvonne Rogers and Siân Lindley. Collaborating around large interactive displays: which way is best to meet? *Interacting with Computers*, 16(6):1133–1152, 2004.
- [70] Kathy Ryall, Clifton Forlines, Chia Shen, Meredith Morris, and Katherine Everitt. Experiences with and observations of direct-touch tabletops. In *TABLETOP '06: Proceedings of the First IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, pages 89–96, Washington, DC, USA, 2006. IEEE Computer Society.
- [71] Hans Sagan. *Space-Filling Curves*. Springer-Verlag, Berlin, Germany, 1994.
- [72] J. Alfredo Sanchez, Guadalupe Quintana, and Antonio Razo. Staf-fish: Starfields+fish-eye visualization and its application to federated digital libraries. In *Workshop on Perspectives, Challenges and Opportunities for Human-Computer Interaction in Latin America (CLIHCO7)*, Rio de Janeiro, Brazil, September 2007.
- [73] Manojit Sarkar and Marc Brown. Graphical fish-eye views of graphs. In *CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 83–91, New York, NY, USA, 1992. ACM.
- [74] Eric Scheirer. *Music-Listening Systems*. PhD thesis, MIT Media Lab, 2000.
- [75] Matthias Schicker. Audiophield: A framework for music similarity analysis. Project Thesis, May 2007.
- [76] Stacey Scott, Karen Grant, and Regan Mandryk. System guidelines for co-located, collaborative work on a tabletop display. In *ECSCW'03: Proceedings of the eighth European Conference on Computer Supported Cooperative Work*, pages 159–178, Norwell, MA, USA, 2003. Kluwer Academic Publishers.
- [77] Chia Shen, Neal Lesh, Baback Moghaddam, Paul Beardsley, and Ryan Bardsley. Personal digital historian: user interface design. In *CHI '01: CHI '01 extended abstracts on Human factors in computing systems*, pages 29–30, 2001.
- [78] Ben Shneiderman. The eyes have it: a task by data type taxonomy for information visualizations. *Visual Languages, 1996. Proceedings., IEEE Symposium on*, pages 336–343, 1996.
- [79] Robert Spence and Mark Apperley. Database navigation: An office environment for the professional. *Behaviour and Information Technology*, 1(1):43–54, 1980.
- [80] Ian Stavness, Jennifer Gluck, Leah Vilhan, and Sidney Fels. The music table: A map-based ubiquitous system for social interaction with a digital music collection. In *International Conference on Entertainment Computing (ICEC05)*, pages 291–302. Springer, 2005.
- [81] Anthony Tang, Melanie Tory, Barry Po, Petra Neumann, and Sheelagh Carpendale. Collaborative coupling over tabletop displays. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 1181–1190, New York, NY, USA, 2006. ACM.

- [82] Aaron Toney and Bruce Thomas. Considering Reach in Tangible and Table Top Design. In *TABLETOP '06: Proceedings of the First IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, pages 57–58, Washington, DC, USA, 2006. IEEE Computer Society.
- [83] Marc Torrens, Patrick Hertzog, and Josep Arcos. Visualizing and Exploring Personal Music Libraries. In *ISMIR 2004, 5th International Conference on Music Information Retrieval*, pages 409–415, Barcelona, Spain, October 2004.
- [84] Edward Tse, Saul Greenberg, Chia Shen, and Clifton Forlines. Multimodal multi-player tabletop gaming. *Comput. Entertain.*, 5(2):12, 2007.
- [85] George Tzanetakis and Perry Cook. MARSYAS: A Framework for Audio Analysis. *Organized Sound, Cambridge University Press* 4(3), 2000.
- [86] George Tzanetakis and Perry Cook. Marsyas3d: A prototype audio browser-editor using a large scale immersive visual and audio display. In J. Hiipakka, N. Zacharov, and T. Takala, editors, *Proceedings of the 7th International Conference on Auditory Display (ICAD2001)*, pages 250–254, Espoo, Finland, 2001. Helsinki University of Technology.
- [87] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. In *IEEE Transactions on Speech and Audio Processing*, volume 10, pages 293–302, july 2002.
- [88] George Tzanetakis, Georg Essl, and Perry Cook. Human perception and computer extraction of musical beat strength. In *Proceedings of the 5th International Conference on Digital Audio Effects (DAFx-02)*, pages 257–261, september 2002.
- [89] Fabio Vignoli. Digital music interaction concepts: A user study. In *ISMIR 2004, 5th International Conference on Music Information Retrieval*, pages 415–421, Barcelona, Spain, October 2004.
- [90] Fabio Vignoli, Rob van Gulik, and Huub van de Wetering. Mapping music in the palm of your hand, explore and discover your collection. In *ISMIR 2004, 5th International Conference on Music Information Retrieval*, pages 409–415, Barcelona, Spain, October 2004.
- [91] Amy Volda, Rebecca Grinter, and Nicholas Ducheneaut. Social practices around itunes. In Kenton O’Hara and Barry Brown, editors, *Consuming Music Together: Social and Collaborative Aspects of Music Consumption Technologies*, volume 35 of *Computer Supported Cooperative Work*, chapter 4, pages 57–83. Springer, Dordrecht, The Netherlands, 2006.
- [92] Colin Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.
- [93] Pierre Wellner. The digitaldesk calculator: Tactile manipulation on a desk top display. In *Proceedings of UIST’92, the ACM Symposium on User Interface Software and Technology*, pages 27–33, November 1991.
- [94] Ian Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2 edition, 2005.

Web References

- [@1] Roberto Amorium and Sebastian Mares. LAME MP3 Encoder. Website. <http://lame.sourceforge.net/index.php>, 10 September 2008.
- [@2] Apple Computer Inc. iTunes Music player. Website. <http://www.apple.com/itunes>.
- [@4] Helsinki University of Technology - Laboratory of Computer and Information Science. Self-Organizing Map research. Website. <http://www.cis.hut.fi/research/som-research/>, 13 August 2008.
- [@5] Paul Lamere. Help bring Search Inside the Music to the world. Website. http://blogs.sun.com/plamere/entry/help_bring_search_inside_the, 15 August 2008.
- [@6] Paul Lamere. More fun statistics about last.fm tags. Website. http://blogs.sun.com/plamere/entry/more_fun_statistics_about_last, 10 September 2008.
- [@7] Last.fm, Ltd. last.fm – The Social Music Revolution. Website. <http://www.last.fm>, 19 August 2008.
- [@8] Last.fm, Ltd. last.fm – The Social Music Revolution – API. Website. <http://www.last.fm/api>, 10 September 2008.
- [@9] Last.fm, Ltd. last.fm’s Playground – Most Unwanted Scrobbles. Website. <http://playground.last.fm/unwanted>, 20 August 2008.
- [@12] Microsoft Corporation. Microsoft Surface. Website. <http://www.microsoft.com/surface/index.html>, 15 September 2008.
- [@13] Microsoft Corporation. Visual Studio 2008. Website. <http://www.microsoft.com/emea/msdn/visualstudio/default.aspx>, 8 September 2008.
- [@14] Microsoft Corporation. XNA Game Studio 2.0. Website. <http://msdn.microsoft.com/en-us/library/bb200104.aspx>, 8 September 2008.
- [@15] Microsoft Corporation. Xna.com. Website. <http://xna.com/>, 8 September 2008.
- [@17] Music Genome Project. Pandora Internet Radio. <http://www.pandora.com>.
- [@18] MusicIP Corporation. MusicIP: Your World. Amplified. <http://www.musicip.com>.
- [@19] MuzicForums.com. Muzic Forums – The Music Community. Website. <http://www.muzicforums.com/>, 20 August 2008.
- [@20] MySpace.com. MySpace FORUMS – Music. Website. <http://forums.myspace.com/s/4.aspx?fuseaction=forums.viewsubforum>, 20 August 2008.
- [@21] Natural User Interface Group (~ NUI Group). Touchlib. Website. <http://www.nuigroup.com/touchlib/>, 8 September 2008.
- [@22] Martin Nilsson. id3v2.3.0 - ID3.org. Website. <http://www.id3.org/id3v2.3.0>, 19 August 2008.
- [@23] Nullsoft. Winamp homepage. Website. <http://www.winamp.com>.
- [@24] Frank Patalong. Musik-download-zählung: Einfalltor für Chart-Manipulationen? Website. <http://www.spiegel.de/netzwelt/web/0,1518,538302,00.html>, 19 August 2008.

- [@25] Dirk Peitz. Musikmesse Popkomm: Der Download kann dem Album nix. Website. <http://www.sueddeutsche.de/kultur/artikel/470/86384/>, 19 August 2008.
- [@26] Claudio Pinhanez. The Everywhere Displays Projector. Website. <http://www.research.ibm.com/ed/>, 15 September 2008.
- [@27] Ben Shneiderman. Dynamic queries, starfield displays, and the path to Spotfire. Website. <http://www.cs.umd.edu/hcil/spotfire/>, 12 August 2008.
- [@28] SMART Technologies ULC. SMART Technologies, industry leader in interactive whiteboard technology, the SMART Board. Website. <http://smarttech.com/>, 8 September 2008.
- [@29] STIBCO Software Inc. Customers of TIBVO Spotfire Enterprise Analytics Products - TIBCO Spotfire. Website. <http://spotfire.tibco.com/customers/>, 12 August 2008.
- [@30] Stefan Toengi. AudioGenie Start. Website. <http://www.audiogenie.de/en/index.htm>, 8 September 2008.
- [@31] un4seen developments. Un4seen developments - 2midi / bass / mid2xm / mo3 / xm-exe / xmplay. Website. <http://www.un4seen.com/>, 8 September 2008.
- [@32] Wikipedia, the free encyclopedia. Collaborative Filtering. Website. http://en.wikipedia.org/wiki/Collaborative_filtering, 20 August 2008.
- [@33] Xiph.org. Xiph.org. Website. <http://www.xiph.org/vorbis/>, 10 September 2008.

Appendix A: webtag_extractors.xml

```

1  <?xml version="1.0" encoding="utf-8"?>
2
3  <!-- This file is used to configure the tags for the web_extractors in AudioPhield. -->
4  <!-- It's xml, so fiddle around as you please - but don't brake the scheme. -->
5
6  <!-- use signifances between 1 and 10. -->
7  <!-- These will get scaled to actual significances inside AudioPhield -->
8
9  <web_extractor_list>
10
11  <!-- ##### genres ##### -->
12
13      <web_extractor significance="10">
14          <contains>
15              <word>pop</word>
16              <word>schlager</word>
17              <match>powerpop</match>
18              <word>disco</word>
19          </contains>
20
21          <contras>
22              <any>metal</any>
23              <any>alternative</any>
24              <word>house</word>
25              <word>techno</word>
26              <word>trance</word>
27              <word>punk</word>
28          </contras>
29      </web_extractor>
30
31      <web_extractor significance="10">
32          <contains>
33              <match>rock</match>
34              <any>country</any>
35              <match>hard rock</match>
36              <match>classic rock</match>
37          </contains>
38
39          <contras>
40              <word>pop</word>
41              <word>soul</word>
42              <match>hip hop</match>
43              <match>hip-hop</match>
44              <match>hiphop</match>
45          </contras>
46      </web_extractor>
47
48      <web_extractor significance="10">
49          <contains>
50              <any>alternative</any>
51              <any>indie</any>
52              <any>grunge</any>
53          </contains>
54
55          <contras>
56              <word>disco</word>
57              <match>hip hop</match>
58              <match>hip-hop</match>
59              <match>hip hip</match>
60              <any>electro</any>
61              <word>house</word>
62              <match>dance</match>
63              <word>eurodance</word>
64          </contras>
65      </web_extractor>
66
67      <web_extractor significance="10">
68          <contains>
69              <word>soul</word>
70

```

```

71         <word>funk</word>
72         <any>blues</any>
73         <any>jazz</any>
74     </contains>
75
76     <contras>
77         <word>disco</word>
78         <match>rock</match>
79         <any>electro</any>
80         <word>house</word>
81         <any>dance</any>
82     </contras>
83 </web_extractor>
84
85 <web_extractor significance="10">
86     <contains>
87         <any>metal</any>
88         <word>grindcore</word>
89     </contains>
90
91     <contras>
92         <word>pop</word>
93         <word>disco</word>
94         <match>rock</match>
95         <any>electro</any>
96         <word>house</word>
97         <any>dance</any>
98     </contras>
99 </web_extractor>
100
101 <web_extractor significance="10">
102     <contains>
103         <match>hip hop</match>
104         <match>hip-hop</match>
105         <match>hip hip</match>
106         <word>hiphop</word>
107         <word>rap</word>
108         <word>rnb</word>
109         <match>drum and bass</match>
110         <match>drum n bass</match>
111         <match>drum & bass</match>
112     </contains>
113
114     <contras>
115         <word>disco</word>
116         <match>rock</match>
117         <any>electro</any>
118         <word>house</word>
119         <any>dance</any>
120     </contras>
121 </web_extractor>
122
123 <web_extractor significance="10">
124     <contains>
125         <any>electro</any>
126         <word>house</word>
127         <match>dance</match>
128         <word>eurodance</word>
129         <any>techno</any>
130         <word>rave</word>
131         <word>trance</word>
132     </contains>
133
134     <contras>
135         <match>rock</match>
136         <any>metal</any>
137         <match>hip hop</match>
138         <match>hip-hop</match>
139     </contras>
140 </web_extractor>
141
142 <web_extractor significance="10">

```

```

143         <contains>
144             <word>noise</word>
145             <word>experimental</word>
146             <match>dark wave</match>
147             <word>darkwave</word>
148             <word>dark-wave</word>
149             <word>ebm</word>
150             <match>electronic body music</match>
151             <any>industrial</any>
152         </contains>
153
154         <contras>
155             <word>soul</word>
156             <match>pop</match>
157             <any>metal</any>
158             <match>hip hop</match>
159             <match>hip-hop</match>
160         </contras>
161     </web_extractor>
162
163     <web_extractor significance="10">
164         <contains>
165             <any>folk</any>
166             <match>singer-songwriter</match>
167             <word>reggae</word>
168             <word>ska</word>
169             <word>dub</word>
170         </contains>
171
172         <contras>
173             <word>pop</word>
174             <any>metal</any>
175             <match>hip hop</match>
176             <match>hip-hop</match>
177         </contras>
178     </web_extractor>
179
180
181     <web_extractor significance="10">
182         <contains>
183             <any>punk</any>
184             <match>new wave</match>
185             <any>gothic</any>
186         </contains>
187
188         <contras>
189             <word>pop</word>
190             <any>disco</any>
191             <match>metal</match>
192             <match>reggae</match>
193             <word>soul</word>
194         </contras>
195     </web_extractor>
196
197     <web_extractor significance="10">
198         <contains>
199             <word>classical</word>
200             <word>oldies</word>
201         </contains>
202     </web_extractor>
203
204     <!-- ##### instruments / singers / sound ##### -->
205
206     <web_extractor significance="4">
207         <contains>
208             <word>male</word>
209         </contains>
210
211         <contras>
212             <any>female</any>
213         </contras>
214     </web_extractor>

```

```

215
216     <web_extractor significance="4">
217         <contains>
218             <any>female</any>
219         </contains>
220
221         <contras>
222             <word>male</word>
223         </contras>
224     </web_extractor>
225
226     <web_extractor significance="6">
227         <contains>
228             <word>instrumental</word>
229         </contains>
230
231         <contras>
232             <any>male</any>
233             <any>female</any>
234         </contras>
235     </web_extractor>
236
237     <web_extractor significance="7">
238         <contains>
239             <word>minimal</word>
240         </contains>
241     </web_extractor>
242
243     <web_extractor significance="8">
244         <contains>
245             <word>acoustic</word>
246             <word>guitar</word>
247         </contains>
248     </web_extractor>
249
250     <web_extractor significance="8">
251         <contains>
252             <any>piano</any>
253         </contains>
254     </web_extractor>
255
256
257 <!-- ##### mood ##### -->
258
259     <web_extractor significance="6">
260         <contains>
261             <word>downtempo</word>
262             <any>chillout</any>
263             <any>lounge</any>
264             <word>atmospheric</word>
265         </contains>
266
267         <contras>
268             <word>uptempo</word>
269             <word>fast</word>
270         </contras>
271     </web_extractor>
272
273     <web_extractor significance="6">
274         <contains>
275             <word>uptempo</word>
276             <match>up tempo</match>
277             <word>fast</word>
278         </contains>
279
280         <contras>
281             <word>downtempo</word>
282             <any>lounge</any>
283             <any>chillout</any>
284         </contras>
285     </web_extractor>
286

```

```

287     <web_extractor significance="3">
288         <contains>
289             <word>breakbeat</word>
290         </contains>
291
292         <contras>
293             <word>melancholy</word>
294             <word>ambient</word>
295             <word>sad</word>
296         </contras>
297     </web_extractor>
298
299     <web_extractor significance="5">
300         <contains>
301             <word>mellow</word>
302             <word>ambient</word>
303         </contains>
304     </web_extractor>
305
306     <web_extractor significance="5">
307         <contains>
308             <word>playful</word>
309             <word>party</word>
310             <word>harsh</word>
311             <any>loud</any>
312             <any>fun</any>
313         </contains>
314
315         <contras>
316             <word>melancholy</word>
317             <word>sad</word>
318             <word>wistful</word>
319             <word>autumnal</word>
320             <match>fado</match>
321         </contras>
322     </web_extractor>
323
324     <web_extractor significance="5">
325         <contains>
326             <word>melancholy</word>
327             <word>sad</word>
328             <word>wistful</word>
329             <word>autumnal</word>
330             <match>fado</match>
331         </contains>
332
333         <contras>
334             <word>playful</word>
335             <word>party</word>
336             <word>harsh</word>
337             <any>loud</any>
338             <any>fun</any>
339         </contras>
340     </web_extractor>
341
342
343 <!-- ##### age ##### -->
344
345     <web_extractor significance="3">
346         <contains>
347             <any>50s</any>
348             <any>40s</any>
349             <any>30s</any>
350             <any>20s</any>
351         </contains>
352         <contras>
353             <any>60s</any>
354             <any>70s</any>
355             <any>80s</any>
356             <any>90s</any>
357             <any>00s</any>
358         </contras>

```

```

359     </web_extractor>
360
361     <web_extractor significance="3">
362         <contains>
363             <any>60s</any>
364         </contains>
365         <contras>
366             <any>70s</any>
367             <any>80s</any>
368             <any>90s</any>
369             <any>00s</any>
370         </contras>
371     </web_extractor>
372
373     <web_extractor significance="3">
374         <contains>
375             <any>70s</any>
376         </contains>
377         <contras>
378             <any>60s</any>
379             <any>80s</any>
380             <any>90s</any>
381             <any>00s</any>
382         </contras>
383     </web_extractor>
384
385     <web_extractor significance="3">
386         <contains>
387             <any>80s</any>
388         </contains>
389         <contras>
390             <any>60s</any>
391             <any>70s</any>
392             <any>90s</any>
393             <any>00s</any>
394         </contras>
395     </web_extractor>
396
397     <web_extractor significance="3">
398         <contains>
399             <any>90s</any>
400         </contains>
401         <contras>
402             <any>60s</any>
403             <any>70s</any>
404             <any>80s</any>
405         </contras>
406     </web_extractor>
407
408     <web_extractor significance="3">
409         <contains>
410             <any>00s</any>
411         </contains>
412         <contras>
413             <any>60s</any>
414             <any>70s</any>
415             <any>80s</any>
416             <any>90s</any>
417         </contras>
418     </web_extractor>
419
420     <web_extractor significance="3">
421         <contains>
422             <word>classic</word>
423         </contains>
424     </web_extractor>
425
426     <!-- ##### from ##### -->
427
428     <web_extractor significance="3">
429         <contains>
430             <word>USA</word>

```



```

431         <any>america</any>
432         <word>canada</word>
433         <word>canadian</word>
434     </contains>
435
436     <contras>
437         <word>british</word>
438         <word>uk</word>
439         <any>german</any>
440         <any>deutsch</any>
441     </contras>
442 </web_extractor>
443
444 <web_extractor significance="3">
445     <contains>
446         <word>british</word>
447         <word>uk</word>
448         <word>london</word>
449         <match>britpop</match>
450     </contains>
451
452     <contras>
453         <word>USA</word>
454         <any>america</any>
455         <word>canada</word>
456         <any>german</any>
457         <any>deutsch</any>
458     </contras>
459 </web_extractor>
460
461 <web_extractor significance="3">
462     <contains>
463         <any>german</any>
464         <any>deutsch</any>
465     </contains>
466
467     <contras>
468         <word>british</word>
469         <word>uk</word>
470         <word>USA</word>
471         <any>america</any>
472         <word>canada</word>
473     </contras>
474 </web_extractor>
475
476 <web_extractor significance="4">
477     <contains>
478         <word>anime</word>
479         <word>japanese</word>
480         <match>j-pop</match>
481         <match>j-rock</match>
482         <match>jrock</match>
483         <match>j-indie</match>
484         <match>j-indies</match>
485     </contains>
486
487     <contras>
488         <word>british</word>
489         <word>uk</word>
490         <word>USA</word>
491         <any>america</any>
492         <word>german</word>
493     </contras>
494 </web_extractor>
495
496 <web_extractor significance="2">
497     <contains>
498         <word>french</word>
499         <word>france</word>
500         <any>francais</any>
501         <word>chanson</word>
502     </contains>

```

```

503
504         <contras>
505             <word>british</word>
506             <word>uk</word>
507             <word>USA</word>
508             <any>america</any>
509             <word>german</word>
510         </contras>
511 </web_extractor>
512
513 <web_extractor significance="1">
514     <contains>
515         <word>polish</word>
516         <word>poland</word>
517         <any>polski</any>
518     </contains>
519 </web_extractor>
520
521 <web_extractor significance="3">
522     <contains>
523         <word>scandinavian</word>
524         <word>finnish</word>
525         <word>finland</word>
526         <word>swedish</word>
527         <word>sweden</word>
528         <word>oslo</word>
529     </contains>
530 </web_extractor>
531
532 <web_extractor significance="3">
533     <contains>
534         <any>latin</any>
535         <word>cuban</word>
536         <word>mexican</word>
537     </contains>
538 </web_extractor>
539
540 <web_extractor significance="3">
541     <contains>
542         <match>world</match>
543         <any>africa</any>
544         <any>arabic</any>
545     </contains>
546 </web_extractor>
547
548 <web_extractor significance="2">
549     <contains>
550         <any>ruusia</any>
551         <any>ruski</any>
552     </contains>
553 </web_extractor>
554
555 <!-- ##### theme ##### -->
556
557 <web_extractor significance="1">
558     <contains>
559         <word>christian</word>
560         <word>worship</word>
561     </contains>
562 </web_extractor>
563
564 <web_extractor significance="1">
565     <contains>
566         <any>politic</any>
567     </contains>
568 </web_extractor>
569
570 </web_extractor_list>
571

```

Appendix B: Task Schedule

The following is the original task schedule used for the user study (kept in German). The task succession depicted here was modified for every other participant so that the two tests concerned with the Soap-interface came before the ZoomFrame-interface.

AudioPhield: Evaluation

Die Evaluation von AudioPhield zielt auf eine Probandengruppe von sechs Personen ab. Jeder Proband soll vor der Evaluation einen Ausschnitt aus seiner Musiksammlung dem Evaluator zukommen lassen; dieser Ausschnitt sollte zwischen 100 und 120 Titel, die dem Probanden wohl bekannt sind und seinen Musikgeschmack möglichst in ganzer Breite widerspiegeln, umfassen. Die Probanden nehmen als Paare an der Evaluation teil.

Bei jeder Evaluationssitzung filmt eine Kamera gleichzeitig die Probanden und ihre Interaktionen mit der Applikation. Ein abschließender Kurzfragebogen soll nicht implizit erfassbare Fragestellungen beantworten und verifizieren.

I. Teil: Einzelevaluation

Die Probanden sollen zuerst einzeln mit der Applikation interagieren, da wichtige Fragestellungen so besser untersucht werden können. Die Einzelphasen sollen maximal 20 Minuten dauern. Danach geht die Evaluation sofort in die Untersuchung von Paarsituationen über.

Fragestellung:	„Discoverability“ der ZoomFrame-Oberfläche: Inwieweit kann die Oberfläche intuitiv verstanden werden? Welche Merkmale und Möglichkeiten erschliessen sich dem Nutzer sofort, welche sind unauffindbar?
Situation:	Beim Eintreten des Probanden läuft AudioPhield bereits und ist mit den Musikdaten des Probanden, die nach dem wohl besten verfügbaren Verfahren platziert wurden (manuell), bestückt. Die Oberfläche befindet sich im ZoomFrame-Modus; ein ZoomFrame befindet sich bereits auf dem Phield.
Durchführung/Aufgaben:	„0-instruction-run“: Die Probanden werden gebeten, die Applikation auszuprobieren. Es werden weder Aufgaben noch Hilfestellungen angeboten.
Zu beobachten:	Da dies die erste Interaktion mit AudioPhield ist, können Features aus zwei getrennten Gebieten entdeckt werden: <ul style="list-style-type: none"> i) Generelle Features <ul style="list-style-type: none"> - Icons repräsentieren Musikstücke - Songs sind nach Ähnlichkeit angeordnet - TapHold: Anspielen - DoubleTap: Abspielen - TapHold/DoubleTap: Pausieren des gespielten Songs - Hold-Seek: Spulen im Lied - Anspiel-Interaktionen nur im Zoombereich

- Linien verbinden Songs des selben Albums
- VolumeController: Lautstärkeregelung
- VolumeController: Verschiebbarkeit
- ii) ZoomFrame Features
 - Frame beinhaltet FishEye
 - Frames können verschoben werden
 - Frames können skaliert / rotiert werden
 - Frameerzeugung über „Ziehen“ aus den Ecken
 - Frame löschar durch minimieren
 - Text zu Songs im Fadenkreuz im Rahmen

Zeitraumen: etwa 4 Minuten

Fragestellung:	<u>Erlernbarkeit der ZoomFrame-Oberfläche:</u> Inwieweit kann die Oberfläche <i>überhaupt</i> verstanden werden? Welche Features bleiben auch nach kurzen Erklärungen unverstanden?
Situation:	Nachdem die Zero-Instruction-Phase abgelaufen ist, werden alle nicht gefundenen features kurz erklärt. Danach wird der Proband gebeten, ein paar einfache Aufgaben zu erledigen, die abprüfen, ob die erklärten features verstanden werden konnten.
Durchführung/Aufgaben:	<p>Es werden nur die Aufgaben gestellt, deren Gegenstand erklärt werden musste. Aufgaben (parallel zu Checkliste oben):</p> <ul style="list-style-type: none"> - Zeige zwei Musikstücke, die einander ähneln - Spiele ein Musikstück ab, pausiere es und setze das Abspielen fort. Spiel ein anderes Stück kurz an (ohne es ganz abzuspielen) - Suche die Positionen „halb“ und „drei-viertel“ in einem Song - Schalte die Wiedergabe auf stumm, verschiebe das VolumeWidget und erhöhe die Lautstärke wieder - [Zeige zwei Songs des selben Albums... nicht immer gegeben, möglicherweise sehr zeitaufwändig, da für die Trail-Anzeige Songs angespielt werden müssen] - Zeige Text zu einem Song an (-> Verschiebbarkeit, Textanzeige, FishEye). Drehe den Text für maximale Lesbarkeit (-> Rotation) - Lass einen ZoomFrame ein ganzes Genre umfassen (Skalierbarkeit) - Erzeuge und lösche zwei ZoomFrames
Zu beobachten:	War der Proband dazu in der Lage die Aufgabe auszuführen? Waren erneute / weitere Erklärungen notwendig?
Zeitraumen:	bis 3 Minuten

Fragestellung:	<u>„Discoverability“ der Soap-Oberfläche.</u> Wie oben.
Situation:	Das Phield wird in den Soap-Modus versetzt. Sonstiger Zustand bleibt.
Durchführung/Aufgaben:	„Zero-instruction“. Wie oben.
Zu beobachten:	Auffindbare Features: <ul style="list-style-type: none">- Berühren der Oberfläche erzeugt ZoomArea- ZoomArea enthält FishEye- ZoomAreas sind verschiebbar- ZoomAreas sind skalierbar- Zoom der ZoomAreas ist einstellbar- ZoomAreas können vereinigt werden- Roter Ring gibt Interaktionsbereich an- ZoomAreas verschwinden automatisch
Zeitraumen:	etwa 2 Minuten

Fragestellung:	<u>Erlernbarkeit der Soap-Oberfläche:</u> Wie oben
Situation:	Wie oben. Kurze Erklärungen zu nicht gefundenen Features
Durchführung/Aufgaben:	Es werden nur die Aufgaben gestellt, deren Gegenstand erklärt werden musste. Aufgaben (parallel zu Checkliste oben): <ul style="list-style-type: none">- Erzeuge eine ZoomArea- Zeige Artist und Titel zu einem Song an (ZoomArea == FishEye)- Erzeuge eine zweite ZoomArea und vereinige beide durch Verschieben- Lass eine ZoomArea mindestens 50 Songs umfassen (-> Skalierbarkeit)- Stelle den Zoom auf „minimal“ und „maximal“- Entferne alle ZoomAreas
Zu beobachten:	War der Proband dazu in der Lage die Aufgabe auszuführen? Waren erneute / weitere Erklärungen notwendig?
Zeitraumen:	bis 3 Minuten

Fragestellung:	<u>Präferenztest:</u> Welche von den verschiedenen Einstellungen hält der Proband für am sinnvollsten?
Situation:	Phield im Zustand wie nach dem letzten Test. Dem Proband wird erklärt, wie verschiedene Einstellungen geändert werden können. Mit diesem Wissen soll der Proband die Einstellung finden, die ihm am sinnvollsten scheint. Der Proband wird gebeten, seine Kriterien und Gedanken zu artikulieren.
Durchführung/Aufgaben:	<p>Zu verändern:</p> <ul style="list-style-type: none">- Umschalten: ZoomFrame / Soap-Oberfläche- Plateaugröße (3 Werte: Nicht vorhanden, normal, groß)- Verschiebungsintensität (3 Werte: schwach, normal, stark)- Linien zu OriginalPosition an-/abschalten
Zu beobachten:	Einstellungen und Kriterien für diese.
Zeitraumen:	bis 4 Minuten

Fragestellung:	<u>Dateneinflusstest:</u> Inwieweit beeinflusst die Güte der Ähnlichkeitsanalyse (und damit der Platzierung) die Erfahrungen?
Situation:	Dem Probanden werden in zufälliger Reihenfolge Phields vorgeführt, die mit Hilfe von MIR-Daten oder tag-Daten bestückt wurden.
Durchführung/Aufgaben:	Der Proband soll jeweils nacheinander 3 möglichst ähnliche Songs verschiedener Künstler anspielen.
Zu beobachten:	Dauer und offensichtliche Schwierigkeit der Aufgabenerfüllung.
Zeitraumen:	bis 2 Minuten

II. Teil: Paarevaluation

Nachdem der zweite Proband den Einzelteil durchlaufen hat, kommt der erste Proband hinzu. Alle Aufgabenstellungen werden von nun an dem Paar gestellt.

Fragestellung:	<u>Generelles Verhalten:</u> Wie verhalten sich die Probanden, wenn sie gemeinsam mit AudioPhield interagieren sollen?
Situation:	Das Phield enthält nun die Songs von beiden Probanden. Platzierungsgrundlage ist wieder die beste verfügbare Ähnlichkeitsanalyse (wohl manuelle Platzierung).
Durchführung/Aufgaben:	<p>In den ersten 5 Minuten wird keine Aufgabe gestellt (außer dem generellen „Bitte interagiert mit der Applikation“). Danach werden Aufgaben gestellt, die die Kommunikation über Musik fördern sollen</p> <ul style="list-style-type: none"> - Bitte macht Euch mit der Sammlung Eures Partners vertraut - Stell Deinem Partner Deinen Musikgeschmack / Deine Musiksammlung vor - Spiel Deinem Gegenüber 3 Songs vor, zu denen man gut tanzen oder entspannen kann - Finde die 3 Songs, die dem Partner am peinlichsten sein sollten - Spiel dem Partner 2 Songs vor, die dieser nicht kennt, aber unbedingt gehört haben muss („Bildungslücken“) - Suche in der Auswahl Deines Partners 2 Songs, die Du auch gerne hättest
Zu beobachten:	Generelles Verhalten. Werden mehr eigene oder fremde Songs gespielt? Wie häufig werden Songs ganz gespielt, wie häufig abgebrochen? Präferiertes Interface.
Zeitraumen:	bis 15 Minuten

Fragestellung:	<u>Konfliktpotenzial:</u> Inwieweit behindert oder fördert das Interface die simultane, gemeinsame Interaktion mit der Applikation?
Situation:	Phield wie zuvor. Es werden kurze Aufgaben gestellt, die simultane Interaktion, z.T. im selben Areal des Phields erfordern.
Durchführung/Aufgaben:	<p>Simultanaufgaben:</p> <ul style="list-style-type: none"> - Stellt Euch jeweils eine Playlist von Songs auf dem Phield auf, die Ihr zum Joggen/Autofahren/in der U-Bahn/... gerne hören würdet

- Spielt als erster 5 Songs an, deren Interpret mit „S“ beginnt

Konfliktaufgaben:

- (Findet und spielt Songs, die Ihr gemeinsam habt) wenn vorhanden
- Findet Künstler, die in beiden Kollektionen mehr als einmal vertreten sind

Zu beobachten:

Auftretende Konflikte und deren Lösung. Wird überhaupt gleichzeitig interagiert? Entwickeln sich soziale Protokolle?

Zeitraumen:

bis 5 Minuten

Appendix C: Questionnaire

AudioPhield User Study: Questionnaire

Demographics

First name (*necessary to link video and questionnaire*): _____

Age: _____

Gender: _____

Technical experience:

little

1	2	3	4	5
---	---	---	---	---

a lot

Experience with TableTop interfaces:

little

1	2	3	4	5
---	---	---	---	---

a lot

Music listening habits

Please base your answers only on your collection of digital music.

Size of the private music collection: _____ songs

Music listening habits: _____ songs per week

How do you primarily select the songs to listen to:

- ☐ I create playlists explicitly
- ☐ I play one specific album at a time (linear or random)
- ☐ I have my entire collection on „shuffle“
- ☐ I have a strongly reduced subset of my collection on „shuffle“
- ☐ I use „smart playlist“-systems (e.g., pandora, last.fm, ...)
- ☐ Other: _____

Do you use music recommendation systems (i.e., pandora, last.fm, musicIP, ...)?

- ☐ Yes ☐ No

If „Yes“, to what extent do these systems motivate your music selection?

_____ % of the songs I listen to were selected automatically

Experiences with AudioPhield

Please rate to what extent you agree with the following statements:

	disagree		neutral		agree
I generally enjoyed interacting with AudioPhield	-2	-1	0	1	2
I felt in control of the application	-2	-1	0	1	2
AudioPhield reacted sometimes unexpected to my inputs	-2	-1	0	1	2
The similarity depiction seemed sensible	-2	-1	0	1	2
I had problems to read the radial texts	-2	-1	0	1	2
The interface felt cluttered and confusing	-2	-1	0	1	2
I got a good overview of my partner's music taste	-2	-1	0	1	2


Experiences with AudioPhield

Please add here all praise, criticism and ideas below.

Your favorite features of AudioPhield:



AudioPhield's biggest flaws:



General improvement ideas:

