Mapping music in the palm of your hand

a visualization algorithm and interaction concept for browsing music on portable players

van Gulik, Rob

Award date:
2004

TECHNISCHE UNIVERSITEIT EINDHOVEN
Department of Mathematics and Computing Science

MASTER'S THESIS

Mapping music in the palm of your hand

A visualization algorithm and interaction concept
for browsing music on portable players

by
Rob van Gulik

Supervisors: dr. ir. Huub van de Wetering (TU/e)
ir. Peter Bingley (Philips Research)
dr. ir. Fabio Vignoli (Philips Research)

Eindhoven, July 2004

Contents

1- Introduction

Advances in digital media and consumer electronics have made it possible to carry more and more content on small portable devices. From the introduction of the Walkman twenty-five years ago to the current portable hard disk players such as the Philips HDD100 or the Apple iPod, a lot of progress has been made in the areas of sound quality, device functionality and the amount of storage available for your music. Standard 60-minute cassette tapes allowed you to store about 15 songs on a single tape, whereas the current hard disk players can store at least a hundredfold more. While some of the first music players with a hard disk were too large to fit in your pocket, such as the Creative Nomad Jukebox shown in Figure 1.1, the current generation of music players with a hard disk is even smaller than a Walkman. Early digital music players were either too large to be portable, or could only store a small number of songs. The Rio PMP300 (depicted in Figure 1.2) belongs to the second group, since it is only capable of storing about 8 songs – whereas the Creative Nomad Jukebox has enough storage capacity for 1500 songs.

Figure 1.1: Creative Nomad Jukebox

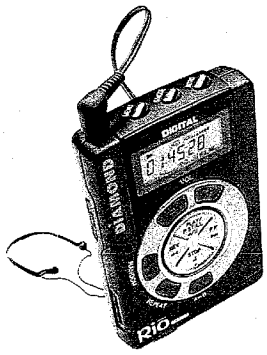


Figure 1.2: Rio PMP300

If you compare these early devices to some of the current digital music players shown in Figures 1.3, 1.4 and 1.5, miniaturization is instantly obvious. Nowadays, portable music players without a hard disk are in some cases hardly the size of a lighter anymore, while still being able to store at least 32 songs - and often a lot more! Music players with a hard disk get smaller as well, bound only by the physical size of the used disk itself. These music jukeboxes are closing in on the point where they are capable of storing your entire music collection – if they are not already there.

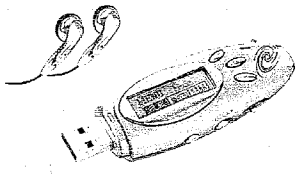


Figure 1.3: CD Cyclone Music Key USB

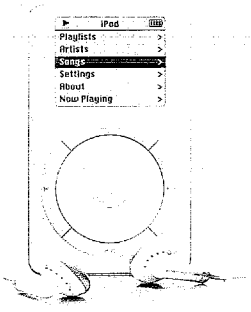


Figure 1.4: Apple iPod

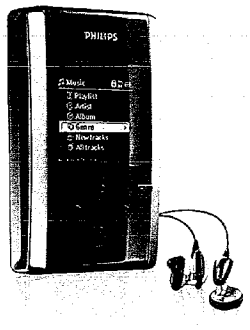


Figure 1.5: Philips HDD100

In fact, investigating the size and storage capacity of the portable music devices more closely, it can be seen that the size of a current digital music jukebox is about the same as the size of an early music player without a hard disk – while the former can store up to 1250 times more songs.

MP3 Player	Size (cm)	Storage Capacity	Storage / cm^3
Rio PMP 300	8.9 x 6.4 x 1.9	32 MB (~8 songs)	0.3 MB (< 0.1 song)
Philips HDD 100	10.7 x 6.4 x 2.0	15 GB (~3750 songs)	112 MB (28 songs)
Apple iPod 40	10.4 x 6.1 x 1.9	40 GB (~10.000 songs)	340 MB (85 songs)

Table 1.1: Comparison of size and storage capacity for portable music players

Although the trends of miniaturization and increasing storage capabilities are generally seen as a good thing – who would not want to have his entire music collection available anywhere, anytime, instead of always dragging along all his favorite tapes and discs – they come with new issues and problems of their own, in particular for user interfaces. Navigating your entire music collection on a pocket-sized display becomes problematic. Especially if the collection contains artists and songs the user is not familiar with, which gets more common every day because of the widespread use of file-sharing programs and online music stores.

Today's portable music players have storage capacity as one of their most important selling points, while very little initiative has been shown so far to make the large amount of content it invites manageable. In fact, most portable music players simply use the traditional directory structure (the kind that PC operating systems have as well) as the interface for their users, with only minimal improvements or even none at all. This thesis focuses on the creation of an experimental user interface for large music collections that fully exploits the available metadata and goes beyond the traditional directory structure. The interface is designed for a future music player; say one that hits the market 5 years from now. In the next sections we first describe the problems we intend to solve with this new user interface, and give an overview of the approach taken. Chapter 2 addresses the problems in more detail by investigating user studies and describing some scenarios, and concludes with a list of requirements for our user interface. We distinguish two categories of requirements: one group regarding the visualization aspects of the interface, and one dealing with the navigation issues. The visualization of a large music collection is discussed in more detail in chapter 3, while chapter 4 tells all about navigation in the user interface. Chapter 5 is about the realization of the interface and the algorithms that have to do with visualization and interaction aspects of the system, and in chapter 6 the user tests are discussed. Finally, in chapter 7 conclusions are stated and some directions for future research are given.

1.1- Problem description

Users of the new generation of portable music players start to face usability problems. Easing the navigation of a large music collection on a small screen is therefore the primary objective of this project. Folder- or list-based structures limit the user to find specific items: an album, song or artist for example. When the user is confronted with a huge amount of digital music, finding the 'right' music for e.g. an occasion or mood can be hard. While it is true that people can customize a group of folders as they wish, it takes a long time to do so for a large music collection. Providing an initial structure based on the hierarchy of genre-artist-album-song is easy – but often not good enough, because the user may not know many of the available items (i.e. artists, albums and songs) and he may not even like to sort music by genre to begin with.

Current product innovations such as the iPod touch-wheel and the super-scroll on the Philips HDD100 are focused on improving the access speed for list-based interfaces. To enable even faster navigation, research has been done on how to find a specific song by whistling its tune or calling out its title.

Figure 1.6: Philips super scroll

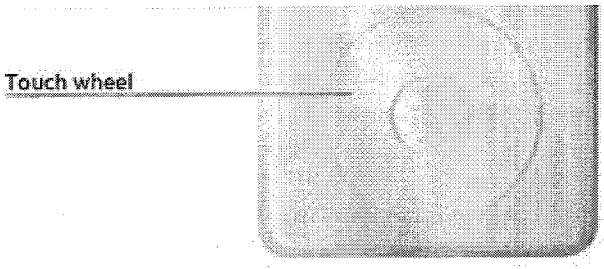


Figure 1.7: iPod touch wheel

In this project, however, the kind of navigation we want to cater for is not only searching for specific songs or artists, but also browsing or exploring an entire music collection. Without doubt the aforementioned innovations improve the ease of use of current interfaces, but unfortunately they do not offer an overview of the music currently loaded on the device. When the users do not know or recognize the name of a specific artist or song, it is difficult to decide whether or not to play the possibly unknown item. Similarly, when they know exactly what song they want to play but do not recall the title, it is very difficult to find it back. And as collections grow, the amount of unknown or forgotten music increases. Consequently, the users end up enjoying only a fraction of the music in their collection. Therefore, the user interface we develop should be capable of providing an instantly clear overview of a music collection and it should facilitate the task of finding music you like when you are not really looking for any song or artist in particular.

Our goal is to enable the users to explore and discover their entire music collection by providing an interactive visualization of the music to be used for browsing and navigation purposes. Apart from designing and implementing the interface itself, we would like to find out whether users actually like it by performing user tests. We hope and expect that users like an interface that helps them find music that suits their current state of mind better than a standard list-based interface.

1.2- Approach

We aim to design, build and test a first prototype of a graphical user interface for music browsing on a small screen. The approach taken for this project can be roughly divided in three phases: first determine what to show to the user (visualization), add functionality later (navigation), and finally test whether users like the interface. An important point we keep in mind when designing this interface is to design for user needs and wishes.

Current user interfaces focus on the standard metadata of genre, artist, album and song, and use this information to present the music collection to the user in the form of a list. Instead of deciding what data to use first and building a user interface on top of that, we turn the process around and decide how to show the information to the user first. The design of our user interface is based on a visualization idea, and we determine what data to use exactly as input for the visualization later. A number of visualization techniques are designed (see chapter 3), and an informed decision is made as to which technique should be implemented.

Designing for user needs and wishes means, in the first place, that we should beware of 'feature creep' – the systematic tendency to load more and more features onto a system at the expense of whatever elegance they may have possessed when originally designed. The user interface should be designed for the needs of the user, and not be loaded with unnecessary features just because they sound nice. When designing an intuitive and easy-to-use interface, less might well be more. However, navigation features that help the user explore his music collection more effectively are encouraged, of course. What it also means is that we should not create an interface based solely on what we *feel* is right, but let the results of previous research and user studies have their influence; we need to listen to what users have to say on the subject. More detailed information gained from user tests – feedback that directly applies to our work – can be used to further improve the design and usability of the interface during development.

Our final implementation of a more complete user interface to be used for user tests will be preceded by a number of early prototypes. During the creation of these prototypes, the focus is on creating the visualization, providing extensive functionality and keeping integration with the existing code-base of the PIC (Personal Infotainment Companion, see section 1.3) project possible. As we will see in chapter 5, the final interface focuses more on design, ease of use and features for navigation. Having an interface that looks nicer and has interesting options while still being easy to use hopefully increases the chance that users actually like the interface and are attracted to play with it.

1.3- Organizational Context

This thesis is part of my graduation project; the concluding project of my computer science Masters study at the Technische Universiteit Eindhoven (TU/e), where I chose the field of computer graphics as my specialization. The project started at September 8, 2003 and is carried out in the Software Architectures group of Philips Research. More specifically, this work is part of the Personal Infotainment Companion project, where the need for an innovative graphical user interface for portable music players with a small screen was identified. The original project description can be found in appendix A. Huub van de Wetering (computer graphics, TU/e) supervised the project together with Peter Bingley and Fabio Vignoli of Philips Research.

1.4- Acknowledgements

Many people have been very helpful during my graduation project. I would like to thank my supervisors, Huub, Peter, and Fabio, as well as Maarten, Albert, Frank, Martin, Arjan, Peter, Boris and all participants in the user tests.

2- Organizing and browsing music collections

The way people organize their digital music collections is influenced by the available metadata. Although people may like alternative concepts for music navigation, they still use standard artist-album-song hierarchies to organize their music. One of the reasons may be that it is easier to use available data then make up your own. In this chapter we take a closer look at the target group – more specifically how they organize and describe music – to find out what kind of metadata they would like to use when organizing and browsing their digital music collections. Music data and metadata is discussed and we briefly compare the metadata with user needs. More thorough investigation of the implications of the needs and wishes of the target group is done in the form of usage scenarios. In section 2.5 the chapter is wrapped up by listing and explaining the specific requirements that the identified issues have lead us to set. These visualization and navigation requirements form the basis of our graphical user interface design. Alongside the requirements we also list our constraints, based on the expected capabilities of future music devices.

2.1- Music organization

People tend to organize their digital music very much like their CD collection: they often keep the same structure for both collections [1]. In general people use catalogue metadata – *artist-name*, *song-name*, *album-name* – combined with some high-level organization such as alphabetical sorting or hot rotation (keeping a special stack or folder with recent or favorite items). The reason is availability; for CD collections catalogue metadata is always available, and for digital music it is provided most of the time as well in the form of tagged data (i.e. ID3 tags in the case of mp3 files) and in the filename itself. In the case of CD collections the physicality of the discs also plays an important role: often played CDs are probably close to the CD player or on top of a stack and visual attributes such as album covers can help in deciding what music to play [1]. For digital music collections these spatial and visual attributes play a much smaller role. Although the standard catalogue metadata are often the only ones used – not only for personal music organization but also in music players such as Windows Media Player [2] and Music Match [3] – people tend to use entirely different terms and expressions to describe the music they want to hear [4]. These descriptions are mostly related to the style, the mood and the genre of the music, or the situation at hand. Some examples of such music descriptions are "Calm music", "Happy music" and "Music for a romantic evening". Other user studies [5] show that people need and like the possibility of using other ways of organizing and browsing their music collections. When confronted with a whole range of features and concepts for music organization and browsing, the participants expressed their interest about the concepts of similarity between songs and between artists.

2.2- Digital music data

The fact that the available amount of storage space on portable music players is still increasing strengthens the need for alternative organization and navigation methods. As we have seen, current mp3 players can store 40 to 60 gigabytes of music already, equaling 10.000 to 15.000 songs. Most of the time these songs have the mp3 format:

> **MP3** (MPEG-1 Audio Layer-3) is a standard technology and format for compressing a sound sequence into a very small file, about one-twelfth the size of the original file, while nearly preserving the original level of sound quality when it is played. The mp3 format is commonly used because of its availability. There are many mp3 players available for most operating systems, and mp3 files can be downloaded from the Internet or created using audio CDs by a process known as ripping and encoding [6].

Some other file formats for digital music files are ogg, wma and wav [7]. Given the status quo of portable music players, the amount of storage space for music on portables five years from now is expected to be at least 100 gigabytes, which amounts to about 25.000 songs. The typical mean size of 4 megabytes per song is a direct consequence of the typical bit-rate used in mp3 compression, which is between 128 and 192 kb/s. The bit-rate signifies the level of compression of the mp3 file: a higher bit-rate (lower compression) means more kilobytes per second and a better quality. At the default bit-rate, one minute of music roughly corresponds to one megabyte of data, resulting in music files that are about four megabytes in size.

A portable music player of the future with a full hard disk will thus contain about 25.000 songs. This high number of items can be hard to organize, either manually or programmatically. But even if a clear automatic organization of the songs is found, we need abstractions to present this data to a user. The small size of the display we have at our disposal (about 10 centimeters diagonally, see constraints in section 2.4) leads to the need for high-level abstractions as well. One possible abstraction is to use artists instead of songs. A small number of tests[1] suggests that the number of songs per artist in a digital music collection of at least 2000 songs is close to 20, meaning that the number of items to visualize drops from 25.000 to about 1250. In the next chapter, which deals with visualization techniques, we will return to this abstraction.

2.3- Digital music metadata

For digital music collections catalogue metadata is easily obtainable and used very often. However, digital music has the potential to offer a lot more information than that provided in tags. This section gives an overview of music metadata, in particular for mp3 files because of the widespread use of this format. Note, however, that our user interface does not depend on the particular format of the music files. We distinguish three different groups of music metadata on the level of individual songs:

- Easily obtainable objective metadata
- Possibly obtainable objective metadata
- Subjective metadata that depends on the user.

2.3.1- Easily obtainable song data

Easily obtainable metadata for music files includes tagged information and some data regarding the physical file itself. Missing tag elements can be retrieved from databases on the web, or from the original CD. Always available or easily obtainable song metadata per song:

- Song title
- Artist name
- Album name
- Year of release
- Length in seconds
- Size in kilobytes
- Sampling rate
- Date created on device
- Time first played
- Time last played

[1] Computing the number of songs per artist in five different music collections of sizes varying between ~2000 and ~5500 songs, we ended up with a mean of 18.9 songs per artist. Note that Berenzweig and Logan did a large-scale evaluation of acoustic and subjective music similarity measures using audio data that closely matches this empirical data: they had an average of 22 songs per artist in their collection [8].

Apart from these data for individual songs, the total number of songs on the device and the corresponding total size in kilobytes are also easily obtainable. The ID3 tag often contains a genre as well. Still, the genre is not included in the list above, because genre specifications in ID3 tags are subjective and depend on the users familiarity with the corresponding kind of music. For example, someone who does not like Rock music may call everything with drums and guitar 'Rock', while someone else may need 20 different genre names to classify the same collection of songs. The metadata list in the following section includes data that is computable from a music file itself. Some of these computations are very easy today, but we decided to leave them in the next section to keep all computable data together.

2.3.2- Possibly obtainable song data

Possibly obtainable music data is objective data that is not readily available, but may be computed or obtained from external sources. Examples are given below:

- Number of times played
- Played until the end last time
- Song / artist is in the top40 (is popular)
- Tempo in beats per minute
- Genre associated with the song
- Mood associated with the song
- Hardness / softness of the song
- Place associated with the song in GPS (global positioning system) coordinates
- Date and time associated with the song
- Other songs / artists associated with the song

The number of times a song has been played and whether or not a song was played until the end last time are included in the 'possibly obtainable' list because this information is not available from the files themselves, but has to be stored by the application instead. Some of the data provided in the two lists above are used for the determination of subjective data. For example, if the system remembers which songs are played until the end and which are not, it may well be able to say something about the users preferences. More examples are given in the next section.

2.3.3- Subjective song data

Taking the user into account, a lot of extra information about songs and artists can be gathered that may be used to personalize the visualization and the interface. User-specific metadata uses some of the data presented above on the level of individual songs. I.e. whether or not a song is in the top 40 can be useful to find out if the user likes top 40 music or not. Examples of subjective music data:

- User likes music of genre x (especially at time / place)
- User likes music of artist y (especially at time / place)
- User likes music of low / medium / high tempo (time / place)
- User likes music from the 60s / 70s / 80s / ... (time / place)
- User likes music of a certain mood (time / place)
- User likes popular music (music in the top 40)
- User likes repetition (playing the same songs again and again)

Apparently, there is a lot of music metadata that can be used for the visualization of music collections and to improve the navigation through them. So far, however, only the most trivial part has been used in actual user interfaces for portable music players: the catalogue metadata, which is the most easily obtainable, including genre information if it is tagged to the music files. The personal descriptions people give when they describe music they would like to hear do not match this catalogue metadata; the metadata is too specific. Attribute information such as the mood and tempo of a song would therefore likely benefit the user. Users would also profit from similarity information on songs and artists, since they expressed their interest in it.

2.4- Scenarios

In this section, several scenarios of using a portable music player are described. The scenarios are based on the functionality of music players and the needs and wishes of the people using such devices. The scenarios are divided into two groups, based on the two actions that are mostly performed with a music player: playing individual songs or start playing multiple songs, for example by using a pre-created *playlist*; a storable list of songs that a user has selected and that are to be played sequentially. Investigation of these scenarios and their implications on preferred features for future music players helps us set requirements for this project. For most scenarios we describe two situations: the case where the user has a current music player with a list-based interface and, if applicable, a more ideal situation in which the user can perform the same action more easily, effectively, or efficiently.

2.4.1- Selecting individual songs

The scenarios described in this section show how people search, select and play a single song.

> **Scenario 1: Jane -** "*Jane just got home with her new portable music player. After uploading her digital music collection of 300 songs – most of which she listens every week – she sits down with the device and carefully selects a song to play. She starts browsing today's newspaper and knows exactly when to return her attention to the device to select a new song, since she knows almost every one of them by heart.*"

Jane will not have a problem using current portable music players. She knows all songs in her collection and knows what she wants to hear – and can easily find it.

> **Scenario 2: Eric -** "*After adding the song collections of both his sons to the music on his mp3 player, Eric sees that there is still room for plenty more songs. He has already uploaded all his own music, some CDs of his girlfriend, and now all this modern dance and R&B stuff of his sons as well. And still there seems to be room for many more songs. After two long nights on Kazaa with the boys, the disk is finally full. Satisfied, Eric starts playing music on the device. As he scrolls through the long list of vaguely familiar names, thoughts like 'I think I heard this name when the boys were watching MTV last week' go through his mind. After having played some of the songs he does know, he starts playing songs randomly, thinking, 'I know there are some calm songs on there – if I could just find out which ones…'*"

After using his mp3-player-with-way-too-many-gigabytes for a few weeks, Eric often finds himself playing songs he does not like at all, for the second or third time. From the list of thousands of songs, he obviously has trouble remembering which he likes and which he does not. He especially enjoys the calm and romantic songs, but there are so many names on there (and a lot of them look or sound quite similar), that he finds it very hard to remember which is which. Removing all music of the boys is not an option, since they started using the device even more than Eric uses it himself. Moreover, his kids may have some nice romantic songs on there as well and Eric may find some of their other music not that bad either. For example, Eric likes happy songs as long as they are not too fast.

Eric knows what kind of music he mostly wants to hear: calm, romantic songs, and now and then some happy music. But having only artist, album and song name to search by, it is hard to find songs that match his preferences – unless he knows exactly which artists and songs are relevant. New music players with a lot of storage capacity allow you to store so much music that knowing every song on the device when it is full gets uncommon or even rare. A visual overview of the music collection in which certain 'kinds' of music are clearly recognizable would be very helpful for people like Eric when searching for songs to play. In the ideal case, Eric can look at his device, click on 'romantic' and get an overview of all romantic songs and artists.

> **Scenario 3: Tanya** - "*Tanya never sits still. She is always busy with something and her music preferences, as well as her mood, change more often than the weather. Tanya knows how to make a playlist, but making a playlist is a very time-consuming activity and she does not have time to waste. Besides, if she would ever make one, she would not like it anymore within the hour. Therefore, she always handpicks her songs, spending only a few seconds each time because there is always something else that needs her attention. Tanya knows most songs on her music player and she always selects a song that matches her current state of mind.*"

Although Tanya knows most songs in her digital music collection, she wants to spend as little time as possible when looking for a song that suits her mood. For people like her, it can be frustrating to have to scroll through a very long list of artists before finally finding something worthwhile. It would be handy for them to have a high-level selection first (i.e. all fast aggressive music), and select a specific song from the resulting subset. That way, Tanya can efficiently find something she likes, no matter how often her preferences change.

2.4.2- Creating and using playlists

The scenarios in this section are about the creation and usage of playlists on portable music players.

> **Scenario 4: Jane** - "*Normally, Jane likes to select her songs one by one, but today she goes jogging. Now a pre-created playlist is handier than having to select songs while on the move. Since she knows and loves all songs on her music player, she does not really care which songs are in her playlist. She has some less specific preferences though: she would like the playlist to last for about one hour, and the tempo of the played songs should gradually increase over this interval.*"

Even though Jane knows all songs, it will take her quite some time to create a good playlist for jogging. Not only does she have to order a number of songs on tempo, she also has to make the step-size just right to ensure that the transition from slow to fast music is made gradually and within an hour. Jane would like her music player to do this work for her: she just wants to tell her player that she wants an hour of music during which the tempo should gradually increase.

> **Scenario 5: Eric** - "*When looking for songs on his full device, Eric notices that he is always looking for songs or artists that are like the songs and artists that he does know. Often, he just wants to play a number of songs that are like the song he is currently playing. Or he wants to hear music that is similar to the music by one or more specific artists, such as Brahms and Puccini, or Jewel and Ilse de Lange.*"

With his current mp3 player, Eric has two options to find music that is similar to what he is currently playing. He either has to listen to some other songs himself and decide whether they are similar, or he can ask other people if they know something similar in the available collection. The ideal music player for Eric would have a button that says 'play similar music' – either for the song that is currently playing or for a number of songs or artists that he just marked.

> **Scenario 6: Tanya** - "*Tanya has had it with her music player. After a few months the scroll buttons started responding somewhat slower, and now it takes her ages just to find a single song! So she decided that creating one or more playlists might be a good alternative. At least it only has to be done once. The first playlist she made simply contained all songs in her collection. When playing it, Tanya clicks next when she does not want to hear the song. She thought about creating different playlists for different moods, but never really made them because it would take too much time. She does not mind pressing next when she hears a song she dislikes, but what irritates her is that she hears the same songs again and again. Even when shuffle is on, the same songs are often played.*"

Creating playlists is a time-consuming task. Tanya, like many other people, does not like to spend much time on this. Creating a playlist should be fast and easy, she says. Tanya wants to create playlists for different occasions without having to spend much time on it. But improving the use of her complete playlist would be a good alternative. I.e. if the playlist remembers when a song was last played, Tanya would not have to hear the same music every day.

> **Scenario 7: Dave** - "*Dave spent an evening creating playlists for his daughter Tanya, listening carefully to her descriptions of the kind of music she likes to listen to at certain times or places. She told him some specific songs and artists, and he added the rest himself. It took a lot of work, but at the end of the day Tanya finally had a music player with great playlists for all sorts of different moods and occasions. However, just a few weeks later Tanya started using her list of all songs again, since the playlists were not accurate anymore.*"

Tanya's preferences did not really change, and the playlists her father made were just about perfect for what she needed. There were playlists for in the bus, for doing homework, for working out, for a romantic evening, everything. The problem is that digital music collections tend to change over time. Not only are new songs added and old ones removed, it also happens that songs are replaced by a version of better quality, or that the song is renamed because its name was not accurate. A playlist is generally just a collection of fixed links to certain songs. As the music collection changes, the playlists get out of date because many links become invalid. Innovative playlist creation methods that focus on the creation of playlists without referring to specific songs may solve this problem.

2.5- Requirements and constraints

Navigation through music collections is traditionally accomplished by using list-based hierarchical structures based on genre/artist/album. Such a structure is only efficient when users want to search for specific artists or songs but does not support the users well when the desires are not specific. The concept of music similarity as well as the personal and vague descriptions about the style, mood and tempo are thus not well suited to use in a list-based user interface. As the size of the average music collection increases, new ways to find desired music in a less specific way are needed. Scenario 2 above shows that providing a clear overview of a music collection may help a great deal when looking for a certain kind of music. In such an overview, elements like similarity, style, mood and tempo can be used more easily since the overview can be designed with these concepts in mind.
Together, the scenarios about selecting single songs presented the following desirable features:

1. The ability to quickly find and select a specific song (scenario 1)
2. The ability to find songs based on less specific requirements (scenario 2)
3. An overview of the music collection that shows what kinds of music are available (scenario 2)
4. Having control over the overview, being able to change it and select subsets (scenario 3)

Creating a playlist on current user interfaces for portable music players is becoming a chore. Finding items you like in the ever-growing and ever-changing lists is hard and takes a lot of time. Moreover, because in most cases many elements are unknown, you miss a lot of interesting songs. Portable music players are devices to be used on the move, and as such, their main functionality should require as little interaction as possible. Currently, creating playlists is done off-site and before actually playing them, because it takes a lot of time. The alternative that people may take is to play the entire music collection as one 'playlist' and just skip to the next song if they do not like the current one. Being able to create a playlist both fast and easy while meeting the users (realistic) demands, even in a large music collection and on a small screen, would thus be a valuable asset on modern portable music players. Based on the second set of scenarios, the following desirable features are added:

5. Automatic playlist creation functionality that builds a playlist given only a small number of parameters, such as the length of the list and the kinds of music it should contain (scenario 4)
6. The ability to find and play similar music, either by song or by artist (scenario 5)
7. Learning capabilities in the device. For example, if the device remembers when songs are last played, it can skip songs. Learning can be made more interesting by making the device remember which songs were played until the end last time (scenario 6)
8. Playlists that are based on kinds of music instead of specific songs (scenario 7)

2.5.1- Requirements

Based on the desirable features from the scenarios and the discussed metadata, we have set the following requirements for the visualization and interaction aspects of our user interface:

Visualization:

- There should be an easy to understand music overview (feature 3)
- Contextual information that makes the overview clear to the user needs to be provided, using easily understandable and familiar music attributes (feature 2, 3)
- Similarity between songs or artists should be used and shown in the overview (feature 6)

Interaction:

- Specific as well as non-specific (fuzzy) searches should be supported (feature 1, 2)
- The user needs to have control over the overview, to navigate his music collection (feature 4)
- The interface should provide an easy playlist creation method, preferably without relying on the availability of specific songs (feature 5, 8)

Note that we have not used the subjective song data in either the scenarios or the requirements. Although some sort of recommendation system that uses subjective song data and learns users preferences would be a nice addition to the interface, it falls outside the scope of this project. Metadata related to learning and subjective metadata, even a basic element such as the time last played (scenario 6, feature 7), will not be used.

2.5.2- Constraints

Given that the user interface we are designing is targeted to accompany a music player about five years from now has the advantage that the technical constraints are not too rigid:

- Graphics capabilities on the device will be at least comparable to the 32 MB ATI M9 chipset
- CPU capacity will be at least comparable to a 500 MHz Pentium3.
- Battery use is not considered
- The device knows its logical location (network location), meaning it knows whether it is at home, at work, in the car, or in a public environment
- Catalogue metadata is always available (this name, artist, album information is available on the internet, so incomplete tags can be filled in)
- Although the device may have some buttons (at least on/off and volume control), controlling the user interface is done using a touch screen
- The touch screen will have a resolution of 640 x 480 and at least 16 bits color (65k), but 24 bits color (16M) is also very well possible
- The size of the touch screen will be about 4 inches (ca 10 cm)

3- Visualizing music collections

Following the approach described in the introduction, the creation of our graphical user interface starts with a visualization technique. Making this visualization interactive and thus creating a user interface with it is performed as the next step. The main reason for this approach is to find out whether starting out from a visualization point of view, as opposed to a human-computer-interaction viewpoint that is mostly taken in this context, can lead to an innovative and useful user interface. There are many ways to visualize large data collections. This chapter starts with a list of potential music visualization techniques, inspired by the data visualization examples found in [9,10]. One of them is picked as the method of choice: the visualization technique that best conforms to the requirements listed in the previous chapter. The last section is about providing contextual information about the contents of the visualization, which improves the clarity of the resulting picture and makes it better usable as the basis for our user interface.

3.1- Visualizations

To visualize a large set of music files we have to answer two important questions: what to visualize (which aspects of the music collection to use for visualization) and how to visualize this data. We have already seen what kind of metadata users would like to have available when browsing their digital music: attributes such as mood, style and tempo, and similarity between songs or artists. Apart from these desired search criteria, people want to find a certain 'kind' of music without having to look for specific items – meaning that the visualization should provide some sort of clear high-level overview.

Screen space is often a major concern when visualizing a large dataset. We have little screen space to begin with and the music collections we want to visualize can be large (25.000 songs). Therefore, making effective use of the available space is of utmost importance. Some techniques that do so are listed below:

- Use 3d graphics, which expands screen space with a virtual third dimension
- Use 2d graphics, but fill and use the available space completely, as done in Sequoia view [11].
- Avoid making everything static; moving objects in and out of your screen space expands screen space over time
- Let the visualization on your screen provide a camera view of only part of the visualized data set, giving the user the opportunity to interactively explore his music collection by moving the point of view

Using one or more of the above techniques to expand your screen space by increasing the information density may have negative consequences on the readability of the visualization. Although the emphasis is on visualization, we need to keep readability and usability in mind. The following guidelines are used to keep the visualization clear:

- Use suitable colors to separate different subsets of data
- Combine coloring with lighting to separate subsets even better
- Use transparency, which makes overlap without occlusion possible
- Provide zooming functionality, so the user can navigate (and change the level of detail)

Part of the screen may be used for other user interface related purposes, but the bulk of the screen space will be used for the visualization of the music collection itself. Some concrete visualization methods that were designed with the aforementioned ideas and guidelines in mind are discussed below.

3.1.1- Music Visualization 1: LOD browsing

The LOD (level of detail) browsing technique is inspired by the traditional list-based interfaces. In a list-based interface, the user essentially walks through a tree of a data collection (i.e. a music collection). Different levels of the tree represent different levels of detail, and the children of a node represent items that belong to the group corresponding to this node. In the case of a music collection, the root represents the entire music collection. Children at the first level represent the music genres. At the second level we have the artists, then albums and finally songs. Instead of listing the names of the genres, artists, albums and songs, however, the LOD browsing technique represents the items in a more visual and artistic way.

In LOD browsing, the first view given of the music collection is at the highest hierarchical level (the lowest level of detail). In this view, the music collection is graphically divided into a number of genres. Selecting one of the genres results in a move to the next level of the hierarchy, one step up in the level of detail: artists. This level is visualized in the same way as the genres, but now showing artists of the chosen genre only. Moving further down the hierarchy we will find albums and finally songs. There are many different ways in which the different levels can be visualized. Examples of applicable visualizations are:

- Tree maps, as depicted in Figure 3.1 (see Sequoia view, [11])
- Traditional list interfaces as we have discussed above
- Galaxies, see Figure 3.2, [12]
- Themescapes, see Figure 3.3, [12]
- Pie charts (circles divided in slices that represent proportions)

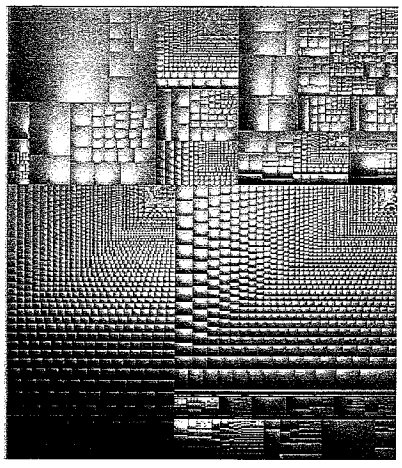


Figure 3.1: An example of a tree map, taken from Sequoia view [11], a visualization technique that makes efficient use of screen space by using and dividing all of it. Hierarchical levels are instantly clear. LOD browsing is possible by navigating to smaller rectangular regions.

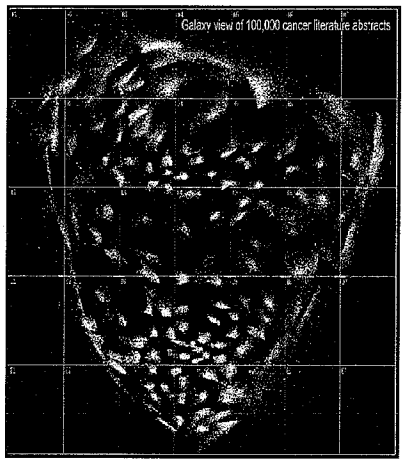


Figure 3.2: An example of a galaxy, taken from [12], a visualization technique in which point clusters suggest patterns of interest. In LOD browsing, selecting a cluster results in a new galaxy with more detailed ('lower level') elements.

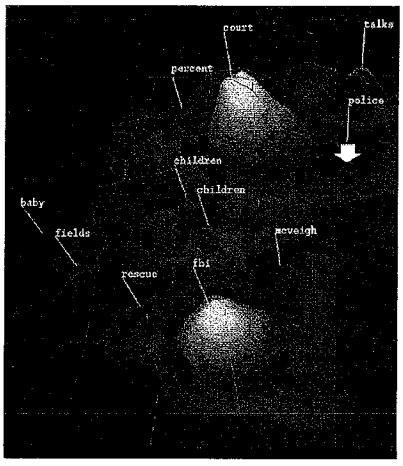


Figure 3.3: An example of a themescape, also from [12], a 3d landscape that exhibits spatial relations revealing information about the visualized items / themes. In LOD browsing, lower levels show a more detailed themescape of the selected region.

No matter what the visualization looks like exactly, LOD browsing can be implemented with the following properties in mind: context should be provided to prevent getting lost, and ease of use should be kept in mind by minimizing the number of mouse clicks required to navigate the music collection. Providing links and a history list can serve these purposes. The history list can provide enough contextual information for the user to know where he is in the hierarchy and the links can help to ease the navigation. The screen setup for the LOD browsing method could for example look as depicted in Figure 3.4.

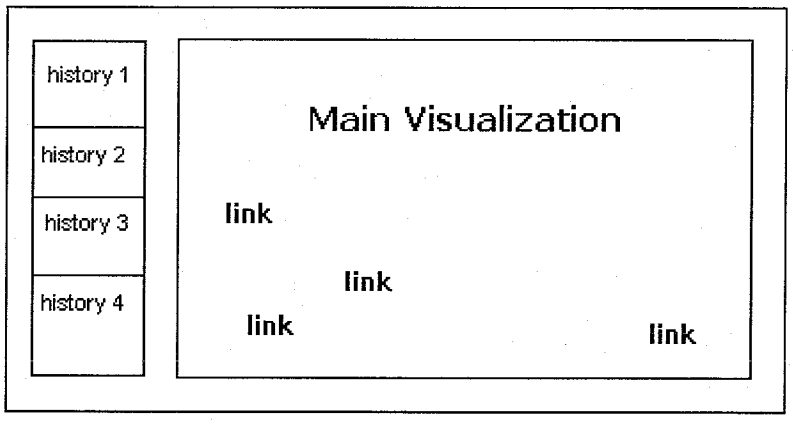


Figure 3.4: Possible LOD Browsing screen layout, with a history list of previous states, and links to other parts of the visualization

History elements of the visualization serve two purposes: they provide a way to go back one or more steps with a single click, and they also show what the previous state looked like at a different level of detail. In the history elements the choice you made at that point could be highlighted, to provide more information and show exactly where you are in the hierarchy – and what the last choices you made were, which resulted in ending up where you are now. Links may be provided in the main visualization, to make it possible to move somewhere else entirely, instead of simply up or down in the hierarchy: to a similar artist for example. The user can be given the ability to create and remove such links himself.

3.1.2- Music Visualization 2: Artist Map

This visualization is based on a graph representation of similar artists. A graph is a set of items connected by edges, where each item is called a vertex and the edges represent a relation on the vertices. Using a large repository of artist information (i.e. an online database), a graph can be constructed in which each vertex represents an artist, and two vertices are connected if the corresponding artists are similar. Whether two artists are similar depends on the information in the database or on some similarity metric. The artist map is a two-dimensional drawing of such a graph. In the map, artists are drawn as circles and connected circles represent similar artists.

Some standard graph-drawing algorithm, for example one that minimizes total edge length or one that minimizes the variance in edge lengths, can be used to determine the initial layout of the graph. A problem with such a standard layout for a potentially large graph is that many vertices and edges may occlude each other, especially if the vertices are labeled and all edges are drawn. An example is shown in Figure 3.5.

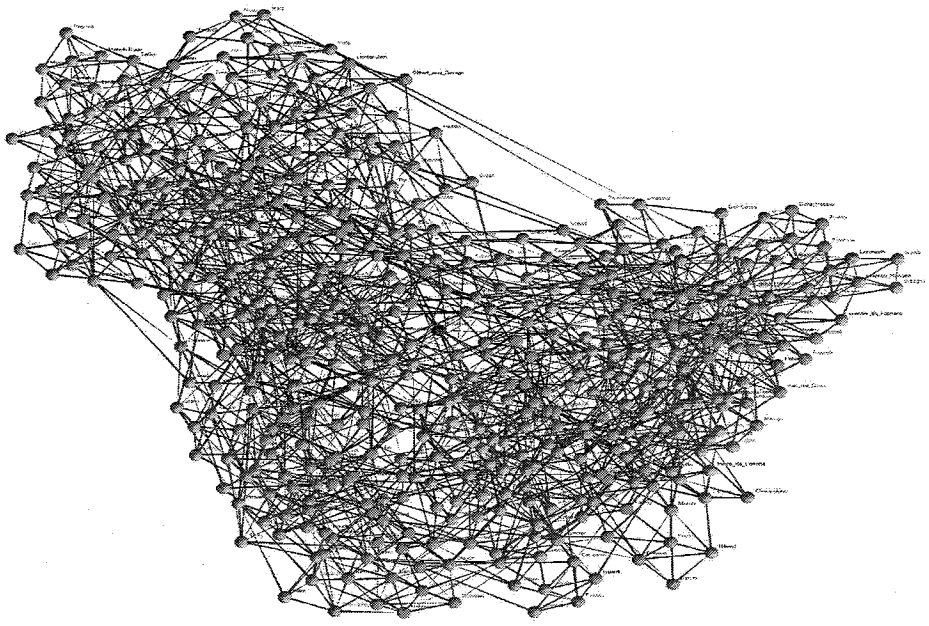


Figure 3.5: A graph in which all edges are drawn and the vertices are labeled. There is too much overlap, and as a result the drawing is unclear [13]

Removing the edges provides us with a much clearer image in the sense that we can see all vertices, but in this image we cannot really see which artists are similar. Vertices that are close to each other may represent artists that are similar, but this depends on the algorithm used to compute the layout. In general, a drawing that only shows vertices does not convey anything about the relations in the graph.

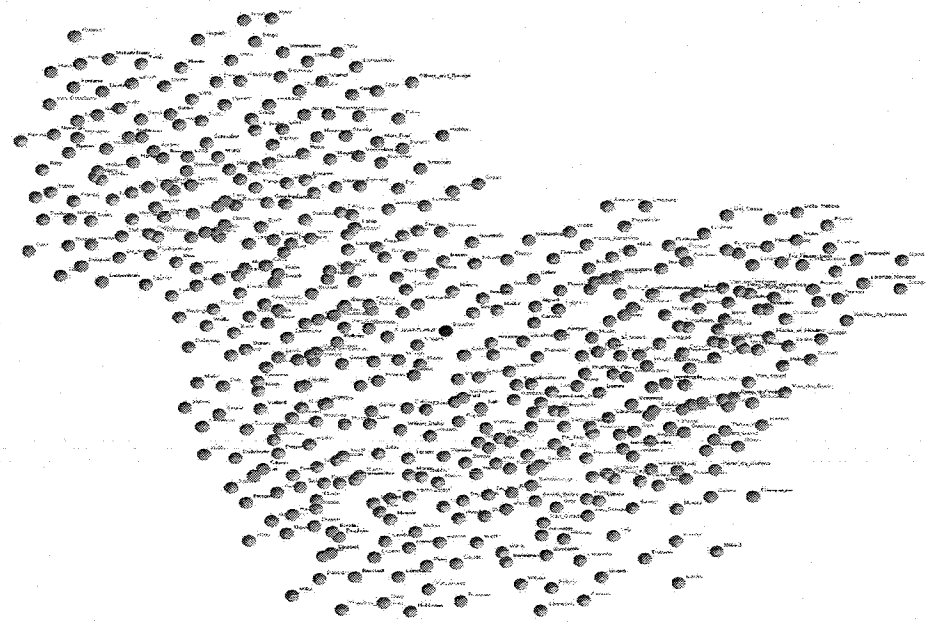


Figure 3.6: The graph from Figure 3.5 with no edges drawn. All vertices are visible, but most information is lost because the edges are gone [13]

In Figure 3.5 there is too much cluttering and in Figure 3.6 most information is lost. If we choose to implement the artist map, we need to find a balance between the wishes: we want to reduce clutter, and still keep similarity information available.

3.1.3- Music Visualization 3: Dynamic Querying

Dynamic querying is a visualization and interaction method based on the VisDB system by H.P. Kriegel [9,14]. VisDB has been developed to support the exploration of large databases. The VisDB system implements several visual data mining techniques, allowing the exploration of large datasets. Figure 3.7 shows the system in action. VisDB makes it possible to explore and navigate a database by using scales and input fields that represent the requirements of the user. Depending on the selections made with these attribute scales and input fields, an overview of all data is given in a picture that contains a single pixel for every item in the database. If the scales and input fields are changed, the picture reflects these changes – hence the name dynamic querying. The picture has the following form: the central pixel represents the most important item, because it satisfies the user requirements at least as good as any other available item. Spiraling outward, the items encountered in the picture satisfy the requirements less and less. Color is also used to indicate the degree to which the corresponding items satisfy the requirements. In figure 3.7, the upper left square is used for visualization of the satisfaction for all attributes together, while the other three squares are used for visualizations depending on only one of the attributes.

Since the display area of our device will be at least 640 x 480 pixels and a full device typically has about 25.000 songs on it (see the constraints from chapter 2), we can display all songs on the display at once. The idea of the VisDB tool can thus be used to visualize music collections on a small screen.

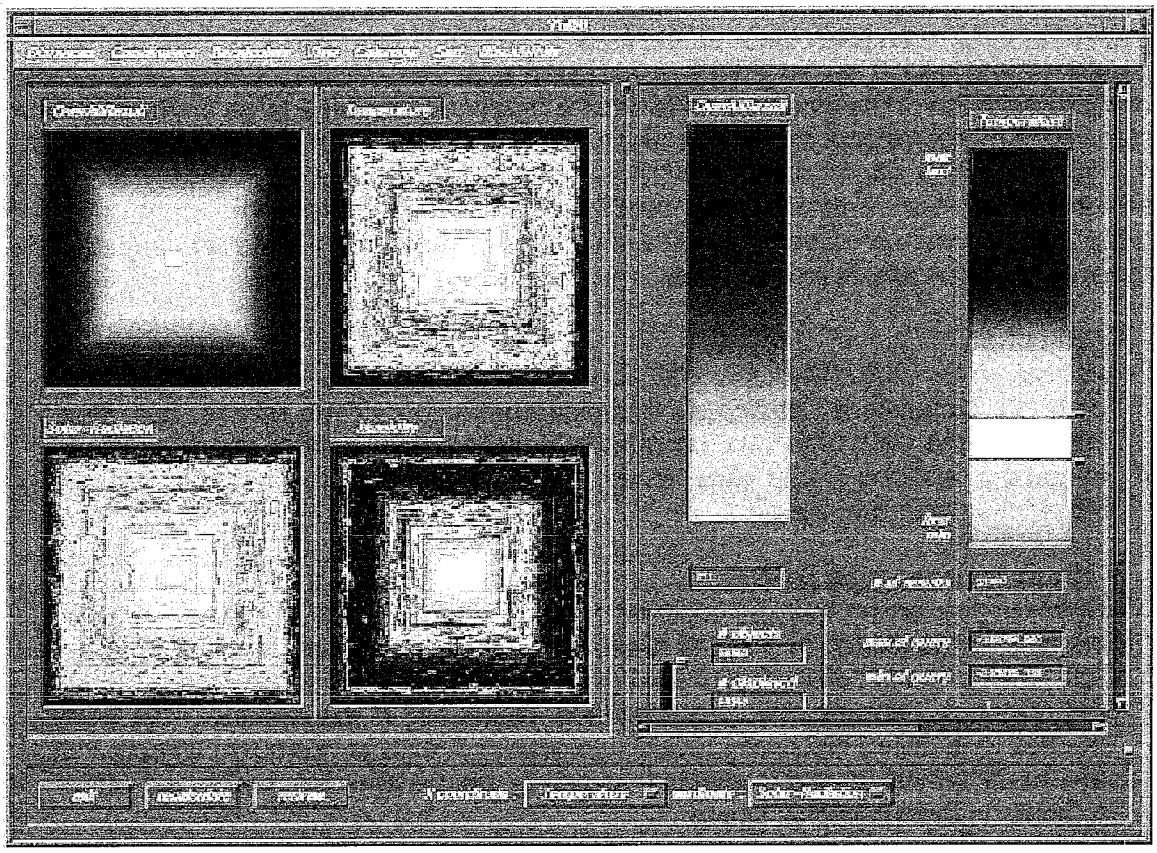


Figure 3.7: The VisDB [9, 14] data visualization tool in action. Selections are made with the attribute scales and input fields on the right, and the squares on the left visualize the items in the collection, sorted on how well they satisfy the selections (starting in the middle, spiraling outward). The upper left square indicates how well they meet all attribute selections together (items in the middle match best), while the other squares correspond to individual attributes

Attributes we could use in a dynamic querying tool for music visualization and navigation are:

- Genre
- BPM
- Year
- Composition (which instruments are allowed / not allowed)
- Loudness
- Hardness
- Song length
- Time since last played

Dynamic querying would be quite useful to get a targeted overview of your music. A problem, however, is that you have to specify what you are looking for before you can see what there is. There is no initial overview of the available items. If you leave all options open, all the squares on the left are colored the same and provide no information at all.

3.1.4- Music Visualization 4: Multidimensional Scaling

The purpose of multidimensional scaling (MDS) is to provide a visual representation of the pattern of proximities (i.e. similarities or distances) among a set of objects. For example, given a matrix of perceived similarities between various brands of a certain product, MDS plots the brands on a map such that those brands that are perceived to be similar to each other are placed near each other on the map, and those that are perceived to be different from each other are placed far away from each other on the map. Note that in this case, the dimensionality of the input data as well as the output data has not been specified: MDS is simply used to create a visual representation of the input, which is often a matrix containing a large amount of numbers. As such, MDS helps people to retrieve and understand the information in this data with less effort. The input data often has a high dimensionality, while the map is mostly drawn in two dimensions. Therefore, in many cases people are unable to interpret the input data directly, while the resulting map is easier to understand. Two ways to map n-dimensional data onto two (or three) dimensions are described below.

Projection
A trivial form of multidimensional scaling is projecting the n-dimensional data onto two dimensions, leaving all other information out of the picture. But often, projecting the data onto two dimensions does not give a clear overview of the important relations in the data, since a large amount of the possibly relevant information is thrown away.

Distance-preserving MDS
Distance-preserving multidimensional scaling refers to a way of representing the information contained in a set of n-dimensional data by a set of points in 2d (or 3d). The points are placed such that geometrical distance between them reflects the distance between the corresponding objects in n dimensions. This MDS technique is called distance preserving because it tries to represent objects that are more similar in n-dimensional space by points that are positioned closer together in 2-dimensional space.

The available metadata for music files have many different dimensions. Trying to visualize all aspects of this metadata at once is therefore very hard. If we want to show as much relevant information as possible, we need to map this n-dimensional data onto two or three of these dimensions. To perform this mapping, we can use the MDS technique. With *projection*, the interface could present the user with a plot of all songs based on two dimensions chosen by default, i.e. genre and year. The user can change the dimensions by selecting them from a list.

An example: the user selects tempo and length of song as the two attributes to make a projection on. He sees some groups, mainly of high tempo with short songs and low tempo with long songs. He then selects a group with relatively high tempo and short songs, and projects this group onto two other dimensions: hardness and genre. From this projection, he selects a group of hard and metal songs and clicks play: all songs in the current selection (high tempo, short, hard, and metal) are enqueued in his playlist. If *distance-preserving MDS* is implemented instead, we end up with one static picture representing the music collection. In this picture, similar songs are positioned close together. 'Similar' in this context is an aggregate function over all available metadata. Possibly, the user can determine which of the metadata are more important and which are less or not at all important – effectively weighing the attributes that are used in the multidimensional scaling.

3.1.5- General visualization improvements

Regardless of the visualization technique we choose to implement, here are some general ideas that can improve the visualization graphically, or improve its usability as an interface:

- **Animations** - Morphing or animating from one level of detail to the next when the user makes a selection or using some other special effect when browsing through the collection does not only make the interface look better; it also increases the ease of use. But, even though animations can increase the usefulness as well as the beauty of the interface, we do not want to bore the user with wasted time between every action. So effects, if available, need to be fast and optional.
- **Recommendations** - Creating interesting recommendations based on user preferences is beyond the scope of this project, but the idea of having a recommendations bar with links scrolling through the screen can still be used. A bar with text-links can show a number of previously played songs, or songs similar to the currently playing song. Clicking a link would result in playing the corresponding song or adding it to the playlist.
- **Album covers** - To increase the fun of browsing, visualizations can be extended with an option to visually leaf through the albums. By showing pictures of the album covers and providing buttons to select the next or previous set, the user can browse through the albums as if they were physically there and select an album based on its visual appearance. In a more basic form, the interface could present album covers of a small number of artists the user is currently looking at.

3.2- Select and refine

The visualization technique that best conforms to our requirements is chosen to be implemented, refined and improved in this section. Important characteristics that the visualization of a music collection should have are the following, as we have seen in chapter 2:

- There should be an easy to understand music overview
- Similarity between songs or artists should be used and shown in this overview
- Contextual information that makes the overview clear to the user needs to be provided, using easily understandable and familiar music attributes

3.2.1- Selecting the visualization

One of the keywords in the requirements is *easy*. We need to keep the interface simple and easily understandable. The **dynamic querying** visualization is not easy to use and understand, even if you know quite well what kind of music your collection contains. Effective browsing with this method can be hard because tweaking the attribute ranges just a little may have large effects on the picture or none at all.

For **distance preserving MDS**, the visualization is not easy to understand either, because the static two- or three-dimensional plot that shows the available information is hard to label. Since all available metadata is used to compute the positions of the elements in the visualization, axes will not have a clear meaning. Therefore, it is hard to provide a clear context in this visualization: the only thing that can easily be clarified is that items that are close in the picture are similar given all the metadata. But for this similarity, the importance or weight of the elements of the metadata has to be set (for example, is the tempo of a song more important than the number of times it has been played? how much more?). This is something we can hardly expect the user of a portable player to do, and as designers we cannot decide what kind of information is more important to most users either. Picking just a few important metadata elements is something we expect people are willing to do, but that would result in projection. Distance preserving MDS will thus not be implemented, as it is not suited for our interface.

Projection is easy to understand and comes with clear contextual information, since the user chooses what the axes mean. Only a basic form of similarity is available in the projection method: dots that are close in the projection represent songs or artists that are similar in the projected dimensions. Unfortunately this 'similarity' does not necessarily say anything about the sound of the music at all, while users would like to have similarity information about the music when they are browsing their collection [5]. For example, users that project on the length of songs and the number of times played may want to know which songs sound similar to the short songs they often listen to. With projection this information is not readily available.

In **LOD browsing**, the overview is simple and easy to understand. The second requirement, however, is not that well covered by this technique. Similarity between artists can be provided in the visualization, but only by giving a large number of links from certain artists to other artists, which clutters the view.

The **artist map** is easy to understand (provided that we solve the cluttering problem) and has similarity as an inherent feature, because similar artists are placed close to each other. Adding contextual information to the artist map is possible by adding labels and colors to the visualization. Since we expect that providing context for the artist map is easy, and the technique thus nicely satisfies all of the requirements, we pick the artist map visualization as the basis for our user interface.

3.2.2- Why Artists

In the artist map, as the name implies, vertices represent artists. Groups of similar artists should be clearly visible in this map. We are working with artists instead of songs here for various reasons:

- Fewer items on the screen means more space per item: we can do something with the visualization of individual items to make them stand out even in the highest level view (i.e. the user can mark interesting artists once and instantly recognize them later)
- We want to ease the navigation of large, possibly unknown music collections by abstracting away unnecessary details. If people know what song they want to play, they can use another method to play this particular song – i.e. by whistling its tune, calling its title, or selecting it in a list. We want to give an overview, and feel that visualizing songs is not necessary in such an overview. Also, visualizing songs may diminish the clarity because of the small amount of space left per item on the small screen
- Fewer items means less computational overhead. Depending on the graph layout algorithm chosen and whether or not the graph is updated dynamically, this can be an important point. Music collections typically have about 20 songs per artist, and this means that many times more processing power is needed to visualize songs. Typically, graph layout algorithms are between $O(n \log n)$ and $O(n^3)$, which means we would need up to 8000 times more processing power to visualize songs [15].

It can be argued that complaining about computational overhead just shows that the artist map does not scale well, since we assume the number of artists to be 'fairly small'. But we do not believe that future increases in storage space will be used for storing a much larger number of music files. Instead, we expect images, music clips and movies to take this space, along with song texts, album covers and other possibly useful information and functionality. The latest trends regarding portable players already show us that the portable machines with a hard disk are no longer only simple music players [16,17,18].

3.2.3- Improving the artist map by sharpening

Although the artist map can meet our demands, there is an information-versus-clarity tradeoff when drawing the graph. The graph contains similarity information in its edges, but drawing the edges clutters the picture, while not drawing them throws away the similarity information. This section shows how the artist map visualization should be changed in such a way that similarity information is available on the screen while the resulting picture is still clear.

To visualize similarity information without drawing edges, similar artists can be drawn so close together that they form visible groups, or *clusters*. If the distance between similar artists is small while the distance between groups of similar artists is larger, a layout is formed that shows which artists are similar without needing edges. The *sharpening* technique by F. van Ham [13] implements this idea. Sharpening accentuates the proximity of neighboring vertices. Figure 3.12 shows an example of the sharpening technique. For more detailed information on graph drawing and layout computation, refer to section 5.2.

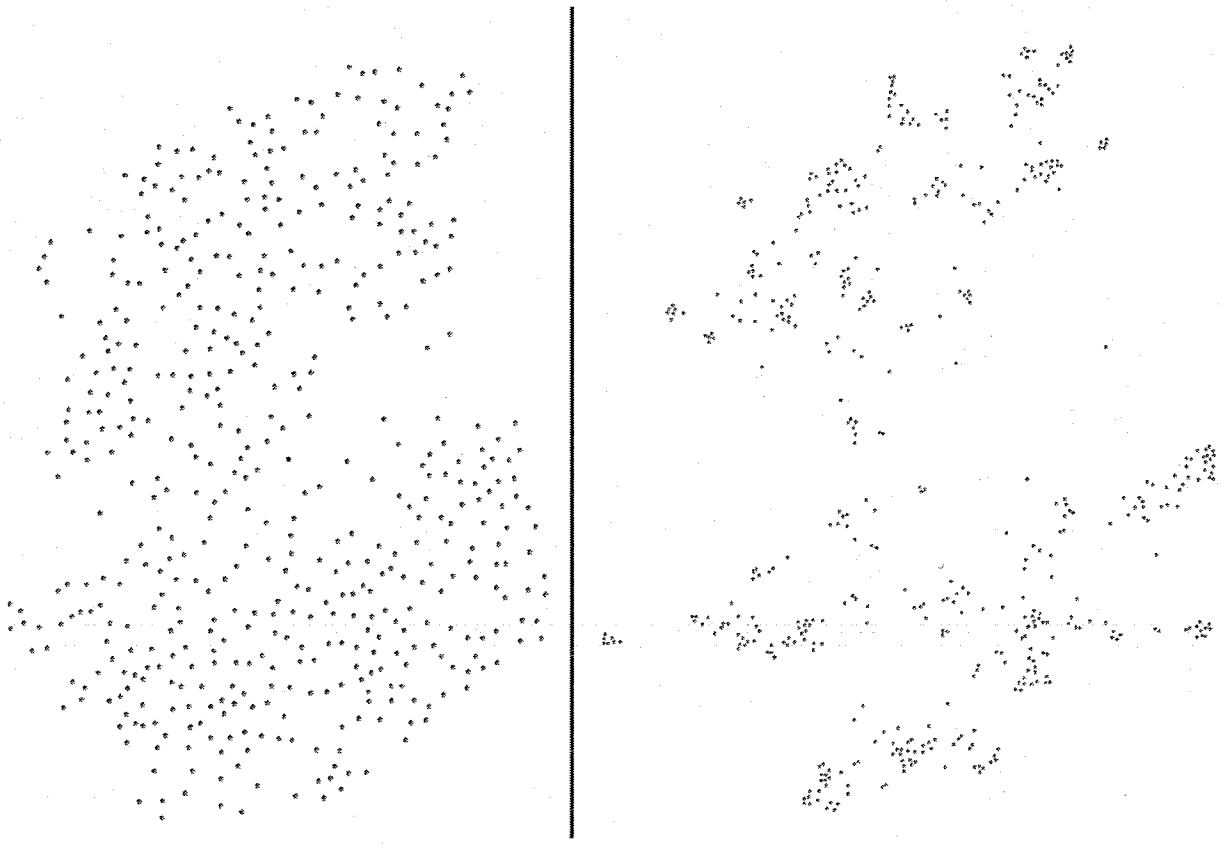


Figure 3.12: Graph sharpening [13]. The graph drawing on the left is generated by a *graph layout algorithm* (see section 5.2) that strives to create edges of uniform length, while the drawing on the right is created by sharpening the drawing on the left

3.2.4- Alternative improvements

Although we will use sharpening to improve the artist map visualization, there are many other ways to arrive at a clear display of information in the artist map. In this section we discuss several of the alternatives that we found less useful than the sharpening method.

2D zooming – showing only part of the graph

To reduce clutter and keep all information available, we could zoom in on part of the graph instead of showing it completely. Moving the viewport shows the neighborhood of the currently shown part of the graph. This way only a partial view of the total number of artist groups is shown, thereby keeping the image clear enough to view without additional effort.

Similarity connections between artists are still drawn as edges between the vertices. One drawback of this approach is that it has an orientation problem: you cannot see where you are in the total graph. Drawing a small view of the entire graph on another part of the screen, highlighting the enlarged part where you are now can solve this problem.

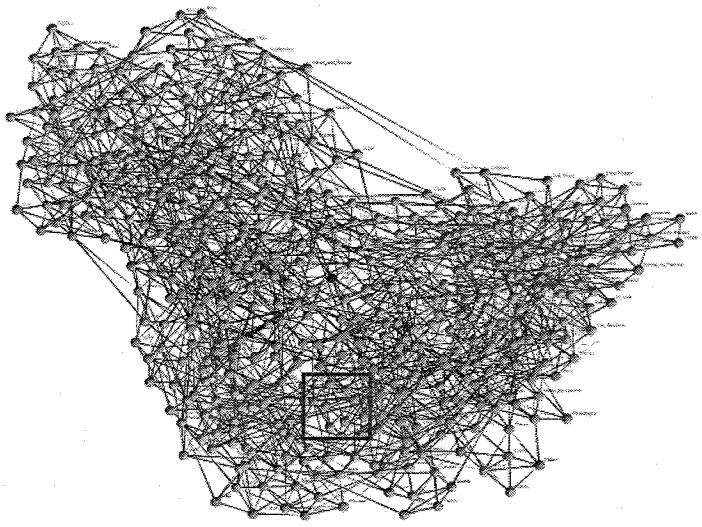


Figure 3.8: A graph with all edges drawn, with a zoom region specified by the rectangle

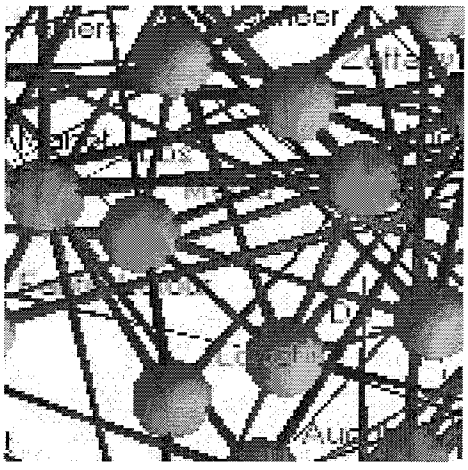


Figure 3.9: The part of the graph on the left after zooming in

But as you can see in the Figures 3.8 and 3.9, enlarging a small part of the graph is not enough to provide a clear view of this part of the graph. Many of the drawn edges are only connected to one or none of the vertices in the enlarged part of the graph. The edges still clutter up the view of the graph. Removing edges that are not connected to any vertex on the screen only improves the clarity a little bit. Thus, zooming does not improve the clarity of the artist map enough.

2D splatting – using color to show vertex density

When you try to visualize large graphs, cluttering due to the large number of objects to display is hard to avoid. Instead of displaying all vertices and edges, the splatting technique by R. van Liere and W. de Leeuw [19] uses a continuous two-dimensional scalar field to represent a graph. Rendering this scalar field (called the splat field) yields pictures that give a clear overview of the structure of the graph, while more detailed information can be gained by zooming in on a region of interest. The usefulness of a splat field is based on the assumption that the density of the vertices resulting from the graphs layout is a meaningful characteristic of the graph. The layout algorithm used to create the graph can make sure this assumption is correct.

If similar artists are drawn close to each other all the time, simply not drawing the edges between them already generates a much clearer picture. Still, the difference between 'close' and 'less close' is hard to see, especially in large graphs. Using the splatting technique, we get an instantly clear overview of similar artist clusters. Zooming in on such a cluster can provide us with more detailed information.

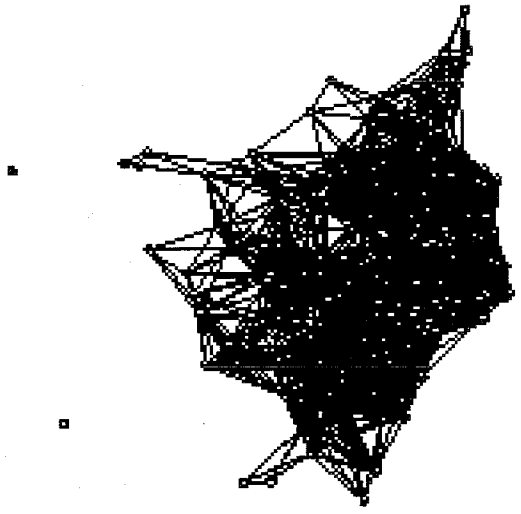


Figure 3.10: A cluttered graph drawing with many overlapping edges. All information is available, but not clearly visible

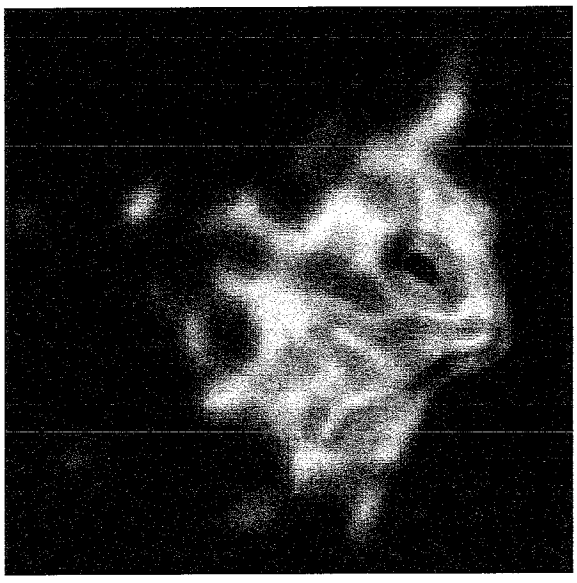


Figure 3.11: A 'splat' drawing of the graph in Figure 3.10 looks much nicer, but vertex density is the only remaining information [19]

The strong point of splatting for artist similarity visualization is that it gives a clear overview of the entire graph, no matter what size, in a single picture. One drawback is that it already uses color to visualize the density of vertices, making it impossible to use color for other purposes such as visualization of an attribute of the music the artists make (i.e. genre). Although intensity could be used instead of color to visualize the vertex density, we expect that the picture may be hard to understand if intensity is used for similarity while color is used for something different.

2D hierarchical clustering

With hierarchical clustering, the layout of a graph contains elements at different levels of detail; vertices can represent individual items as well as groups of items (i.e. items of a higher hierarchical level). A group of items represented by a single vertex is called a cluster. The level of detail of the elements in the graph is varied depending on the *point of interest*: selecting a vertex sets the vertex to be the current points of interest, and if it is a cluster, the vertex expands into a subgroup of more specific elements. After moving the point of interest away from the current group, the vertices in the group form a cluster again. Figure 3.13 shows an example of the hierarchical clustering idea.

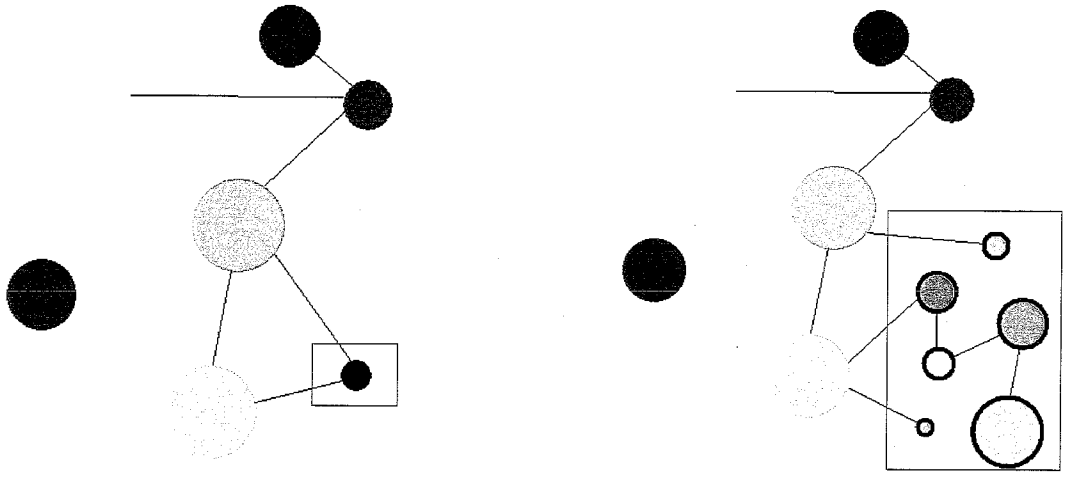


Figure 3.13: Hierarchical clustering example. The picture on the left shows a high level graph with groups of items as vertices (named *clusters*). The rectangle specifies the group to be expanded. Color represents the kind of elements and size represents the number of elements in a group. On the right the same graph is shown after expanding the selected group. Color on the right provides new information; the borders are black to show the kind of the group, but the inside has a color based on the kind of elements within the subgroups. If another group is selected, say the large one on the top left, the subgroups in the currently expanded group form a single vertex (a cluster) again.

In the case of music visualization, the used hierarchy is that of genre-artist-album-song. In the first view of the music collection, the vertices in the graph represent different genres, and two vertices are connected if there is a connection between any artist in the first genre and any artist in the second genre. When selecting one of the genres, the vertex expands to a sub-graph of artists, and the total graph still shows the other genre vertices as well. Selecting an artist vertex also results in expanding the vertex, showing all albums on which this artist performs. When selecting a vertex of a hierarchical level that is already open (selecting another genre, or selecting another artist), all expansions below this level are undone. For example, if Rock and a rock artist are selected first, and Dance later, the expansion of the artist vertex to all of their available albums and the expansion of Rock into all Rock artists are undone, and the genre vertex of all Dance music on the device is expanded into a sub-graph of Dance artists.

To preserve space, vertices far away from the point of interest can be grouped as clusters as well. For example, if a genre contains many artists, groups of artists far away from the currently selected one may be grouped as clusters even though they reside in the currently expanded genre. The farther away vertices are from the point of interest, the larger the clusters get. Frank van Ham has implemented this technique in his hierarchical graph drawing research [20]. Since having a single point of interest and a strict hierarchy limit the navigation through the graph, we chose not to implement hierarchical clustering for the artist map.

3D visualizations

All of the above ways to visualize an artist similarity graph can be easily transformed into a three-dimensional visualization method. Instead of projecting the graph visualization on a plane, it can also be mapped on a 3d volume. The advantage of doing so is that you have another dimension to use for visualization purposes: height can get a useful meaning. For example, the height of part of the graph may visualize the number of songs in the corresponding category.

Two disadvantages of visualizing a graph in 3D are the need for more graphics processing power and the introduction of occlusion: small items that hide behind larger ones are difficult to find, and large items may block your view of the rest of the music collection. For this study we choose not to use a three-dimensional visualization; we stick to two-dimensional visualizations in order to avoid the inevitable 3d occlusion problem.

3.3- Creating context

So far, the artist map visualization consists of a collection of circles on a plane, in which circles positioned close to each other represent similar artists. An example is shown in Figure 3.14.

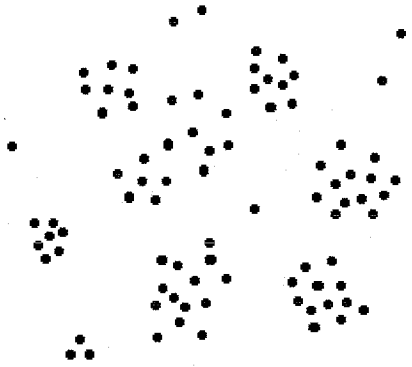


Figure 3.14: Sketch of the artist map. Dots that are close to each other represent similar artists, but it is not clear what kind of music a given cluster contains

Although it is clear in Figure 3.14 that certain artists are similar, for example the three artists at the bottom left of the picture, the artist map lacks contextual information at this point. A *label* near this group with the word 'Nederlandstalig' would make the picture a lot clearer. At the very least it would tell us that the artists in the lower left region of the map make songs of which the lyrics are in Dutch. Another thing it would convey is that there are probably only three artists on the device that sing Dutch songs. Such labels, simple text strings on top of the map, would already provide useful context. Other ways to create more contextual information in the map are the usage of *color* and *size*. Color can be used to represent the value of a certain attribute related to the music: i.e. fast music could be colored red. Other attributes could be used for coloring as well: Dutch songs could be colored orange. Sad songs could be colored blue. Songs that were uploaded last week could be colored pink. As long as the coloring scheme used is clear to the user (providing a legend will do), color can be helpful to provide extra context. The size of the circles that represent artists can be used to show how much music of the corresponding artist is available on the device. Figure 3.15 shows another sketch of the artist map, which makes the added value of contextual information provision instantly clear.

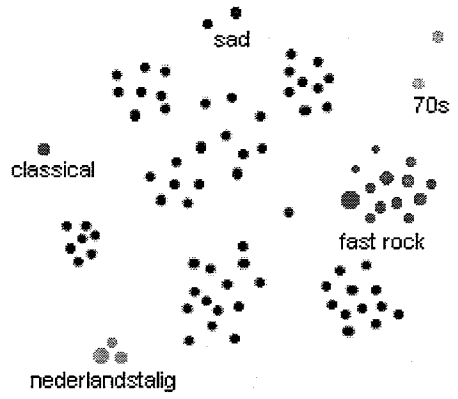


Figure 3.15: Another sketch of the artists map from Figure 3.14. Size, color and labels can provide useful context. Coloring needs a legend so it can convey information different from what the labels show. I.e. instead of coloring fast rock red, newer and older fast rock songs may get different colors

Although only some of the artists in Figure 3.15 are colored and only a few labels are placed, the picture clearly shows the positive effects coloring and labeling have for the artist map. However, the picture also shows a problem: placement of artists seems to be arbitrary. There is no reason to have sad artists at the top of the layout. Note that graph drawing algorithms may produce vastly different layouts for only marginally different inputs. This means that after adding new songs to the device, the sad label and the corresponding artists may end up at the bottom of the graph instead. Such an arbitrary and inconsistent placement is hard to understand. In the next chapter we describe a way to solve this problem and guarantee that labels can always be placed intuitively.

4- Interaction and Navigation

The map of similar artists, as described in the previous chapter, can visualize a large music collection on a small screen. But using this visualization to present a clear overview is just a starting point: fast and easy navigation through the music collection can be added by defining a user interface on top of the visualization. This chapter discusses the browsing and interaction aspects of the artist map, which are based on the interaction-related requirements from chapter two – repeated here for ease of reference:

- The user needs to have control over the overview, to navigate a music collection
- Specific as well as non-specific (fuzzy) searches should be supported
- The interface should provide an easy playlist creation method, preferably without relying on the availability of specific songs

We have seen that placing labels (informative text strings) and coloring the artists can be good ways to clarify the artist map visualization. However, to make effective use of labels and coloring we need to make sure that the positioning as well as the coloring of artists is easy to understand. In section 4.1 we explain how the positioning of artists in the map was made intuitive as well as user-controlled, and in section 4.2 the coloring scheme we use to distinguish certain kinds of music are discussed. Section 4.3 describes how users of the artist map can personalize their view on a music collection. Section 4.4 is about zooming in the artist map and in section 4.5 we present a novel way for creating playlist. Although we focus on the use of the visualization for non-specific searches, we also want the interface to support specific ones, as provided by traditional list-based interfaces. Therefore we designed an integrated interface in which users can seamlessly navigate through their music using the method they prefer. Some advantages of integrating both navigation methods in one interface are described in section 4.6. Visualizing artists instead of songs in the visualization influences several of the interaction aspects described here. Throughout the chapter the effects of this choice will be discussed.

4.1- Artist positioning

In this section we discuss how we changed the artist positioning process in such a way that labels can always be placed intuitively and the user can influence where artists are positioned.

4.1.1- Using magnets

A layout of a music collection gives for each artist a position on the map. In our case, the positioning of artists is based on similarity between artists, so a good layout puts similar artists near each other. The user can visually group these similar artists together. To improve the clarity of the map, labels are used that show what kind of music a certain group of artists makes. Unfortunately, it can be hard to name these labels. Moreover, music collections tend to change frequently, leading to changes in the layout. Labels therefore seem to get an arbitrary position, making it hard if not impossible for a user to memorize the gist of the layout – or even where a single kind of music can be found. To prevent disorientation of the user we try to stabilize the layout by introducing fixed points, called *attribute magnets*.

Attribute magnets are used to roughly define a priori the desired position in the layout for certain kinds of music. The magnets are actively involved in the creation of the layout, so that artists are positioned close to the labels that match their music. Some advantages of using magnets are:

- Labels get a fixed position and are easily nameable (since we simply label the magnets)
- Certain kinds of music get a fixed region in the map, making the overview easier to use
- The user has control over the visualization, since he can control the magnets

4.1.2- Magnet types

We have seen that style, mood and tempo are often used when people describe the kind of music they would like to hear [4]. We use four magnet types to describe style, mood and tempo:

- **Genre** - a category describing the kind of music a song or artist belongs to
- **Year** - the year of release for a song, or the period of activity for an artist
- **Mood** - a term referring to the state of mind a song is associated with
- **Tempo** - the mean number of beats per minute (BPM) in a song

With these magnet types, we expect that users will be able to find the music they would like to hear, as they can use their personal description almost directly as the search criterion. We will now discuss these attribute (magnet) types more thoroughly, and list the exact magnets we use.

Genre and year together capture the style of music: a classification that expresses the kind of music a song or artist belongs to. Genre is particularly appropriate to use for classification of music unknown to the user, because genre can give a useful clue about the kind of music to expect. Year is needed in addition to genre, since genre alone does not always capture the style: R&B in the seventies is different from R&B now. To reduce the number of different year values, ranges are used. An artist can have any distribution of year-range values, although most will fall in only one or a few neighboring ranges. Genre is unique for each artist in our case, as will be explained in the next chapter. The genres and year-ranges used are listed in Table 4.1. The number of genres and year-ranges is intentionally kept low because we do not want to have too many labels on the small screen. Seven genres and six year ranges have been chosen that suit the music collection we use for testing. In our collection most music is from the 90s, which results in a large group of artists surrounding the 90-00 year magnet. In a final product such ranges may be changeable or adapt automatically to fit the available music collection.

Mood is a term describing the emotional state of mind of the listener. Songs are assigned a certain mood – the mood associated with the sound of the song – and artists can therefore have different distributions of mood values. A band may have 10% sad and 90% happy songs for example. The different moods we distinguish are listed in Table 4.1. The list of moods is taken from Moodlogic, a music classification and navigation system for the pc [21].

Tempo is related to the beats per minute of a song. Instead of providing exact BPM ranges, only five tempo classes are discerned – see Table 4.1. The reason is that terms such as 'slow music' are easier to understand than quantifications such as 68 < BPM ≤ 80, and we expect that in general people like the easy labels better. An artist can have any distribution of tempo values.

Attribute type	Attribute magnets
Genre	*Rock, Alternative, Soul/R&B/Rap, Americana, Popular, Dance/Lounge, Nederlandstalig*
Year	*Before 1960, 60-70, 70-80, 80-90, 90-00, 2000+*
Mood	*Aggressive, Brooding, Sad, Sentimental, Mellow, Romantic, Happy, Upbeat, Unknown*
Tempo	*Very Slow, Slow, Medium, Fast, Very Fast*[2]

Table 4.1: Magnets for the attribute types used in the artist map

An important requirement to be able to actually use the attribute magnets to attract artists is that we need to know what kind of music these artists make. Conceptually, artists are attracted to the magnets that describe the kind of music they make. But to do that, attribute information about the music these artists make is needed. In the next chapter we discuss various possible sources for such information, and explain which sources were used for this project.

[2] In BPM: very slow ≤ 68, 68 < slow ≤ 80, 80 < med ≤ 120, 120 < fast ≤ 155, 155 < very-fast. These ranges are based on the results of perceptive studies. Most mainstream music (i.e. most of our test-collection) has a medium tempo

4.1.3- Default magnet positioning

Although magnets can be used to give the user control over the visualization, they should also provide an easily understandable initial view. Therefore, magnets get a default positioning such that the initial view shows a useful high-level picture even though there is no prior knowledge of user preferences available.

The most important requirements for the default positioning of the magnets are as follows:

- The magnet positions should be such that the resulting layout is easy to understand
- Screen space should be used efficiently

In the subsections below we first describe the default positioning for each of the magnet types individually, and then discuss the placement of magnets when more than one type is used.

Positioning a single magnet type

For each of the four magnet types, this section describes the default placement.

Tempo is categorized in five groups: very slow, slow, medium, fast, or very fast. The tempo attribute has a natural ordering that can be mapped to for instance the y-coordinate, such that artists with slower music have a lower y-coordinate. To make efficient use of screen space, we do not place the tempo magnets on a straight vertical line, but instead map the x-coordinate of the tempi as shown in Figure 4.1.

Mood has nine different elements: upbeat, happy, romantic, mellow, sentimental, sad, brooding, aggressive and unknown. Mood has no natural ordering, but looking at the moods more closely we see that moods can be mapped on two axes: positivity and tempo. Common sense tells us that upbeat and aggressive songs are mostly songs with a high tempo, while mellow and sentimental music is mostly slow. In fact, user studies [22] have shown that slow tempo songs are often associated with sad or sentimental feelings, while up-tempo songs are associated with feelings of excitement. Apart from the tempo scale in the different mood elements, we can also identify a scale from negative to positive in the moods. We map tempo on the vertical axis and positivity on the horizontal axis, as shown in Figure 4.2. In addition to meeting the requirements of clarity and efficiency, this layout choice has an added benefit: when switching from mood magnets to tempo magnets or vice versa, the distance artists have to move to fit in the new layout will often be small.

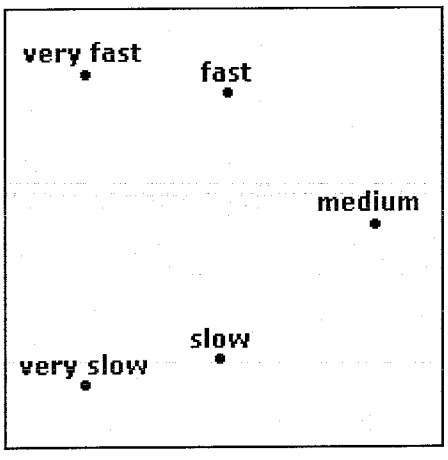


Figure 4.1: Default placement for tempo magnets, when no other magnets are used

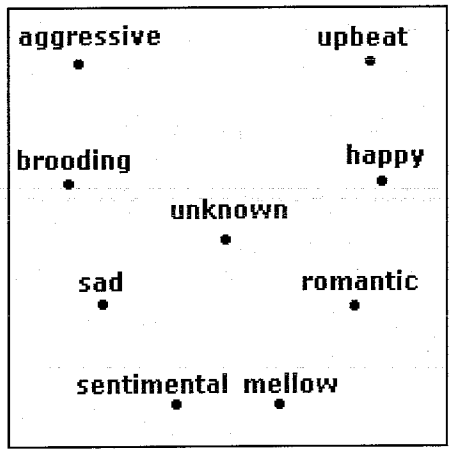


Figure 4.2: Default placement for mood magnets, when no other magnets are used

Genre has seven different elements: Rock, Alternative, Soul/R&B/Rap, Americana, Popular, Dance/Lounge, and Nederlandstalig. Genre has no natural ordering. To meet the efficiency requirement, we position the genres in a circle, using most of the available screen space. The default layout we use for genre is shown in Figure 4.3.

Year has six ranges: before 1960, 60-70, 70-80, 80-90, 90-00, and 2000+. Year has a natural ordering again, and we map time on the horizontal axis. The vertical axis is used to allow for better space-filling properties. See Figure 4.4. The 90-00 magnet is given more space than the other magnets because our test music collection contains more music from the 90s than from any other year range. An improved default magnet positioning would take the actual number of songs for each year range on the device into account and place the magnets accordingly.

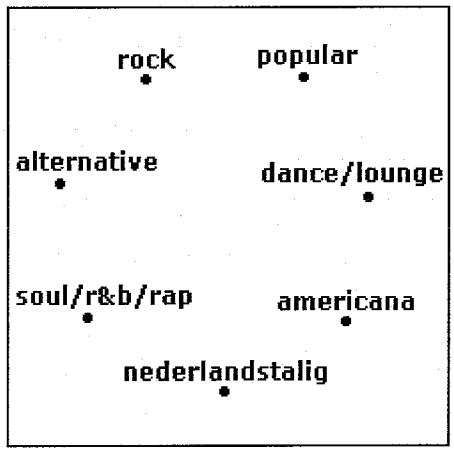


Figure 4.3: Default placement for genre magnets, when no other magnets are used

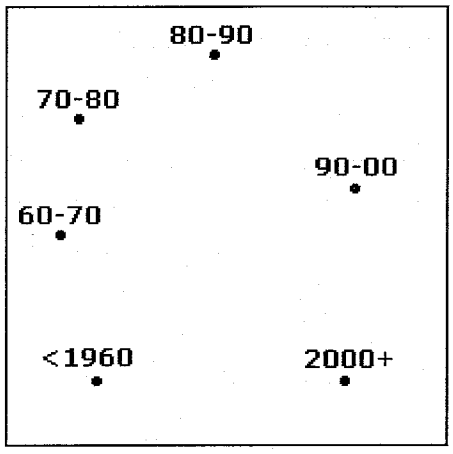


Figure 4.4: Default placement for year magnets, when no other magnets are used

Positioning multiple magnet types

The default magnet positions described above are appropriate when the user selects only one magnet type to use. However, the user decides which magnet types are involved in the creation of the layout. He may use no magnets at all, or choose to use two magnet types. If no magnets are used at all, artists are positioned close together if they are similar and there are no fixed points. If two magnet types are combined, the influence of each individual type should be obvious in the resulting layout. The default positioning proposed above does not work well in this case – the magnets should be placed such that their influences are easily distinguishable.

To this end, when two magnet types are combined, magnets of one kind are placed horizontally and magnets of the other kind are placed vertically. Moreover, the first magnet type only influences the positioning of artists horizontally, while the other only influences the positioning vertically. Tempo and year magnets have a natural order that we can reuse in this context: tempo is mapped on the vertical axis and year is mapped on the horizontal axis.

Since mood and genre do not have a natural ordering, it does not matter whether they are mapped horizontally or vertically. For example, if genre is combined with tempo, genre is placed horizontally and if genre is combined with year, genre is placed vertically. If genre and mood are combined the positioning does not matter, as long as one of them is placed horizontally and one is placed vertically. Figure 4.5 shows how mood and genre magnets are positioned per default when two magnet types are combined. For mood, the positivity is mapped either horizontally or vertically and for genres there is no order: the positioning we chose is arbitrary.

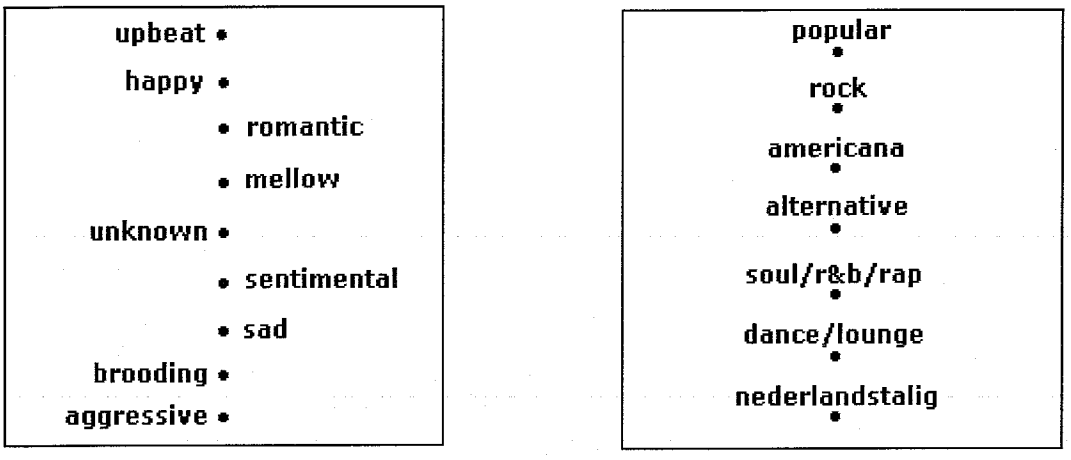


Figure 4.5: Default placement for mood (on the left) and genre (on the right) magnets if another magnet type is used as well. Horizontal placement is also possible, depending on the other magnet type in use. For example, if mood is combined with tempo, mood is placed horizontally because tempo is always mapped vertically.

Combining three or more attribute types for magnet attraction is not allowed, because it is hard if not impossible to position three magnet types in such a way that the layout resulting from the magnet placement is easily understandable. Instead of using three magnet types at once, the user can use two magnet types first, zoom in on a region of interest and use a different magnet type in the resulting subset of the music collection. Zooming is discussed in section 4.4.

4.1.4- Attribute ranges

In the chosen visualization of music collections, *artists* are positioned in a two-dimensional map depending on the attributes of the music they make. The user determines what attributes are used for the placement of the artists, by selecting and positioning the magnets we described.

The fact that we chose to use artists as the basic elements in our visualization leads to some difficulties, because attribute values are generally defined for songs and not for artists. An artist can make music that fits different magnets of the same type (i.e. slow as well as fast). For example, we can say that the song *Metal Militia* from *Metallica*'s first album is a very fast and aggressive song. It is from 1983, and the associated genre is Rock. But this does not make Metallica a very fast and aggressive artist from 1983. Metallica does not only make very fast songs, and not all of their songs are aggressive either. Furthermore, they are still making albums, so the year of release of their music spans a period of over 20 years.

So artists represent a whole range of values for each attribute – except for genre, which is assigned to an artist instead of a song. These attribute ranges leave us with a choice for magnet attraction: either attracting artists to the most dominant attribute only, or attracting them to all matching attributes for the appropriate percentages. For example, say that Metallica has twelve songs in our collection. Four of those songs are slow and eight of them are fast. Given the default tempo magnet-positioning scheme in Figure 4.1, where do we expect Metallica to be positioned? Figure 4.8 shows two options.

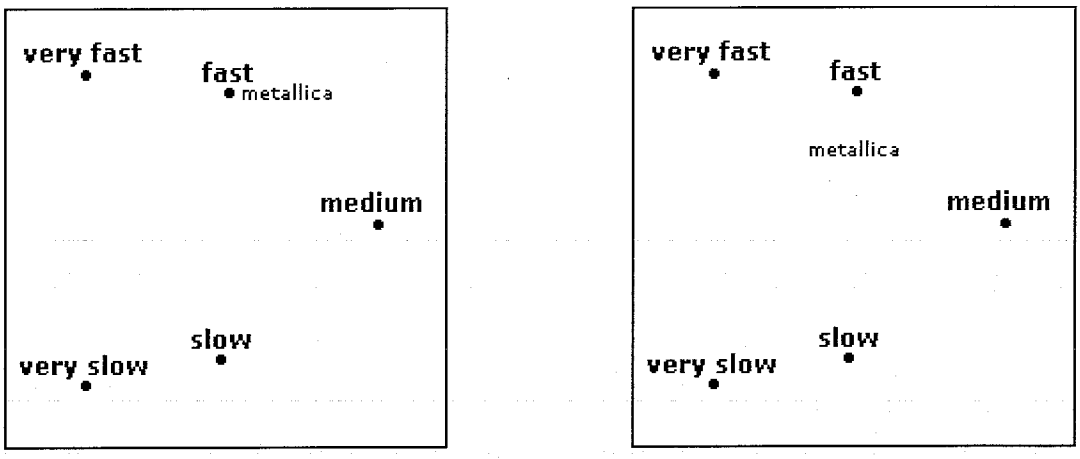


Figure 4.8: Positioning of Metallica in the artist map: should this artist be positioned near the fast magnet because their music mainly has a fast tempo, or should it be positioned between fast and slow because they have some slow songs as well?

In this case, we may be tempted to say that the picture on the right is better. But what if we have four songs by another artist in the collection, two of which have a very fast tempo and the other two have a medium tempo? Metallica and the other artist would get a position that is nearly the same given the magnet positions, while the reason for ending up at their respective positions is very different. Add more artists and more magnets and the picture gets even harder to understand[3]. For this reason, we use the *dominant attraction* rule:

> **Dominant attraction** – for each magnet type, artists are only attracted to one magnet: the one that matches his songs best and describes the largest part of his available songs.
>
> **Partial attraction** – for each magnet type, artists are attracted to all relevant magnets, but only partially: the attraction strength depends on the ratio of matching songs (the number of songs by this artist matching the attribute divided by the total number of songs for this artist)

An obvious drawback of dominant attraction is that when you look at artists near the slow magnet, you only find artists with mostly slow songs – you do not find any of the few ballads by a metal band for example. However, the artist map is designed to provide an overview of a music collection and to make it easier to find a certain kind of music when you are not looking for specific songs or artists. The fact that you only find artists of which you mainly have a given kind of music and none of the outliers (i.e. one slow song by a speed-metal band) may not really be a problem. If users like a certain artist and want to find out more, they can find all their music using the list interface.

There are other ways to solve the problem of the attribute ranges, but all of them have disadvantages of their own. Some possible solutions are listed below.

- **Positioning artists based on partial attraction**, as in the right hand side of Figure 4.8. Metallica for example is attracted to the fast magnet for 67 percent and to the slow magnet for 33 percent. As we have seen, partial attraction can have negative consequences on the clarity of the picture.

[3] Although most artists do not have a highly diverse repertoire, this situation often occurs because people tend to have only a small subset of the songs by any particular artist. I.e. even though 28 out of a total of 30 songs by artist X are fast, if the user only has two songs by X, a fast one and a slow one, it appears as if half of his songs are slow

- **Visualizing sub-artists**, splitting artists if they make multiple kinds of music. The problem with this approach is that one artist may have a large number of sub-artists, especially when multiple magnet types are used. For example, there are 33 Bjork songs in our test collection, which span 8 different moods and 4 different tempo groups. If mood and tempo magnets were used together, this would result in 32 different Bjork artists in the worst case (i.e. songs of every tempo for every mood). Having one artist at many different positions on the screen can be confusing to the user.
- **Visualizing songs instead of artists**: songs are only associated with one attribute per type. However, we decided to visualize artists instead of songs for several good reasons (in chapter 3).

The dominant attraction approach is used because it is the easiest. Not only for users to understand, but also to implement. Zooming is also easier to understand when dominant attraction is used than it is in the case of partial attraction, as section 4.4 will illustrate. Although dominant attraction has the drawback of not helping users find the 'other music' that artists make, it remains to be seen whether users even mind when they are looking for a kind of music.

4.2- Attribute coloring

Artists in the visualization are graphically represented by a circle of which the color depends on a configurable attribute type. In the following we explain, for each attribute type, on what colors the attribute values are mapped. Our attribute coloring for the artist map is based on:

- User studies by A. Cruts and H.C.M. Hoonhout [22]
- Correlation experiments in a test music collection

The user studies described in [22] investigate whether there exists a consistent coloring of music, and if so, what attributes of the music people base their color choices on. The studies have shown that music genre is not likely to provide a reliable and consistent basis to determine the color of music. **Tempo**, however, had a significant influence on the colors people chose. Slow tempo songs are associated to cold colors (such as purple, blue and green), while songs with a high tempo are associated to warm colors (red, orange and yellow). Based on these results, we chose colors for the tempo attribute as shown in Table 4.2.

Tempo value	Associated color
Very slow	Dark blue
Slow	Light blue
Medium	Green
Fast	Yellow
Very fast	Red

Table 4.2: Coloring of the tempo attribute, based on user studies

The report also mentioned that slow tempo songs were associated to sad feelings, and the same was true for cold colors. Up-tempo songs were said to be associated with feelings of excitement, just like warm colors. Therefore, we will color the **moods** upbeat, happy, aggressive and brooding in warm colors, and romantic, mellow, sentimental and sad in cold colors.

To visualize the difference between the more positive moods upbeat, happy, romantic and the more negative ones such as aggressive, brooding, sad and sentimental, we color the positive moods brighter than the negative moods. Mellow is a more neutral mood, and is associated to a neutral color such as brown, or green. The colors we chose for mood are shown in Table 4.3.

Mood value	Associated color
Upbeat	Bright red
Happy	Bright orange / yellow
Romantic	Pink
Mellow	Bright green
Unknown	White
Sentimental	Blue
Sad	Purple
Brooding	Dark orange / yellow
Aggressive	Dark red

Table 4.3: Coloring of the mood attribute, based on user studies

Since **genre** cannot be linked directly to colors, we determined the color associated to a genre by looking for correlations between tempo and genre and between mood and genre. This way, genre could be colored based on the tempo or mood that matches best. The correlation tests were performed on a collection of 2248 songs, and the results are shown in appendix B. Based on these experiments, the colors we chose for the genre attribute are shown in Table 4.4. Note that although the experiments may have resulted in a reasonable genre coloring, the coloring we present here is rather arbitrary and may not reflect the user's personal preference.

Genre value	Associated color
Rock	Bright red
Alternative	Bright orange / yellow
Soul / R&B / Rap	Pink
Americana	Bright green
Popular	White
Dance / Lounge	Blue
Nederlandstalig	Purple

Table 4.4: Coloring of the genre attribute, based on correlation experiments

Year has a natural order, but since fast as well as slow music, happy as well as sad music, and music of all sorts of genres is made every year, there is no clear coloring associated to this order. Therefore, we color the year attribute gray and only vary the brightness, see Table 4.5.

Year value	Associated color
Before 1960	Black
60-70	Very dark gray
70-80	Dark gray
80-90	Medium gray
90-00	Light gray
2000+	White

Table 4.5: Coloring of the year attribute: the future is bright

4.3- User-defined context

In the initial state we choose to use only the genre magnets, since people often classify their music by this attribute. For coloring, the tempo attribute is chosen. By mixing an often used music categorization attribute with less often used tempo information, we hope that on one hand users feel at home, while on the other hand they are triggered to explore the data.

After getting acquainted with the default view on his music collection, the user is expected to experiment with the coloring and magnet attraction options, thus changing the context to reflect his current needs. He can change the attribute type used for coloring or the magnet types used for positioning. Furthermore he can move the magnets on the map. By doing so, the user creates a custom configuration that shows a personalized view of the music collection. Personal configurations can provide a valuable method to quickly scan a music collection for interesting music: if the user can use the configuration on another collection, the created overview instantly shows whether there is anything interesting available given the fuzzy search criteria encoded in the configuration. Two sketches of the artist map are shown in Figures 4.6 and 4.7.

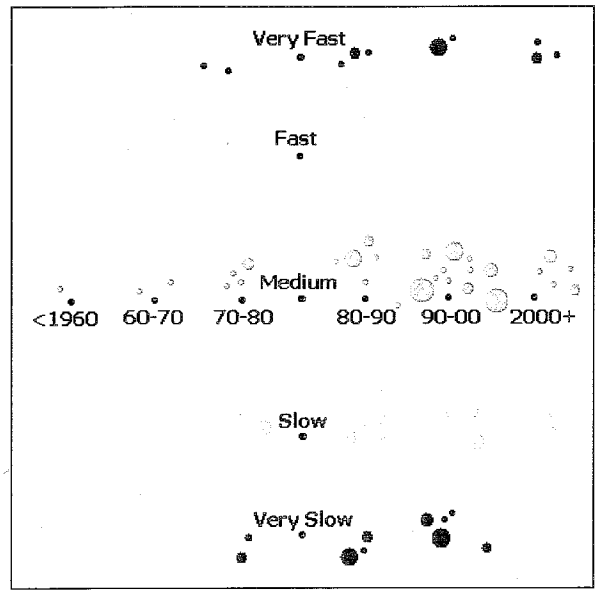


Figure 4.6: An example of an artist map in which year and tempo magnets are used. Year magnets attract artists horizontally, and tempo magnets attract artists only vertically. Artists are colored on the tempo attribute. Larger circles represent artists of which more songs are available on the device

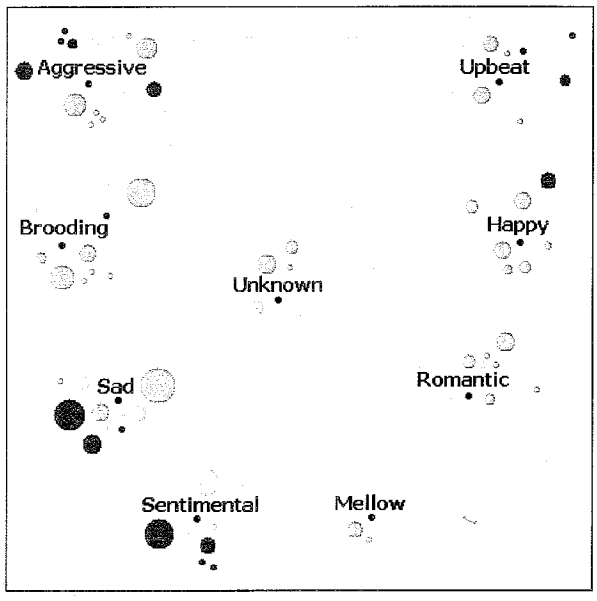


Figure 4.7: An example of another artist map. In this map, mood magnets are used, which attract artists both horizontally and vertically since there are no other magnets in use. Artists are colored on the tempo attribute. Larger circles represent artists of which more songs are available on the device

The artist map places similar artists close together, and uses magnets to ease the navigation and provide fixed points in the layout. Note that if the user changes the magnets that are used, he also influences the similarity relations portrayed in the graph. For example, if three artists sound similar, but two of them make fast music while a third one makes slow music, the third artist is not positioned near the other two artists if tempo magnets are used – while all three of them may be positioned close together when other magnets are used instead.

4.4- Zooming and selecting

So far we have focused on the users interaction with, and control over, the visualization of the entire music collection. But apart from providing an overview of the entire collection, the interface should also enable the user to explore a region of this music map in more detail. The initial overview of the music collection shows the highest-level view of the collection; every artist on the device is visualized. Zooming in on a certain position on the map we get a more detailed view of a subset of the collection: the part that corresponds to the given zoom-position. Such a zooming action is called a *semantic (geometrical) zoom*.

Zooming in, or visually selecting a subset of the music collection, and zooming back out are thus features we would like the interface to have. Although zooming by itself was not a specific requirement, it does help in giving the user control over the visualization, and it is a valuable tool when combining specific and non-specific searches, as we will see in section 4.6. Before describing the design of the zooming action, the effects of attribute ranges on zooming are discussed below. In this discussion we will consider both dominant and partial attraction because of the interesting complexity that partial attraction adds to zooming.

4.4.1- Zooming and attribute ranges

Using the dominant attraction rule simplifies our zooming design, because it allows geometrical zooming, whereas partial attraction does not. If partial attraction were used, zooming would be hard to describe and hard to understand. Figure 4.9 shows an example artist map with a selected region to zoom in at.

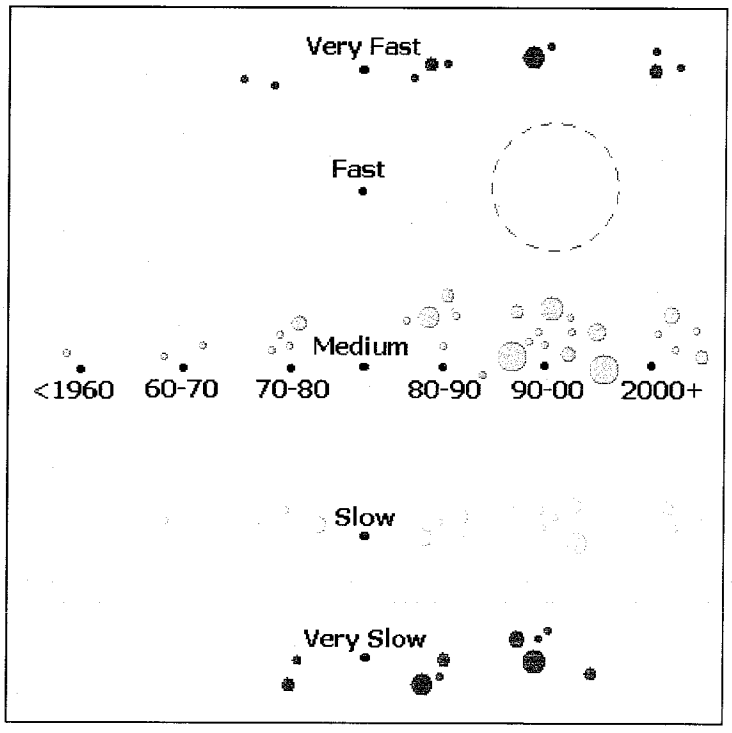


Figure 4.9: The artist map from Figure 4.6, in which year and tempo magnets are used. Year magnets attract artists horizontally, and tempo magnets attract artists vertically. Artists are colored on the tempo attribute. Larger circles represent artists of which more songs are available on the device. The dashed circle shows the zooming region, containing artists with fast music from the 90s

In Figure 4.9, the zooming region intuitively contains artists with fast music from the 90s. Considering the dominant attraction rule that we use, this is indeed correct. However, if we used the partial attraction scheme, there could be artists within this zooming region that do not have any fast music at all, but only some very fast and some medium tempo songs. Along the same lines, artists with songs from the 80s and some brand new songs could also end up in the same zoom region. Apart from creating hard to understand layouts in the high-level overview, partial attraction introduces even more problems when it comes to zooming. Say an artist ended up in the zoom region shown in Figure 4.9 because there are some medium and some very fast songs by this artist on the device, but no fast songs. After zooming in, it is hard to predict what users would expect. Showing an artist of which you do not have any fast songs after zooming in on your fast music can hardly be called intuitive, but making an artist disappear after zooming in right on top of it may be even worse. After a zooming action, the most logical result is a screen showing a scaled up subset of the original picture. This is the result of a standard *geometrical zooming* action and we would like our zooming to be like this, or equally intuitive.

For partial attraction, geometrical zooming does not work, because it results in a screen with artists that do not conform to the kind of music you would expect to find. Instead, the zooming action would need to make a selection of the artists based on the search criteria encoded in the selected zoom region. In this case, 'zooming' would be a subset selection method on the attribute ranges that match the selected region. As we said before, such a counterintuitive zooming method is hard to explain to users, and it is even harder for a user to understand what happens if he uses the action without prior explanation.

Dominant attraction has the advantage that for all artists near a magnet there are songs on the device that match the corresponding attribute. A disadvantage of dominant attraction in the context of zooming is that you are not guaranteed to find artists with fast and very fast songs by zooming in between those magnets. An artist with fast as well as very fast songs is either positioned near the fast magnet or near the very fast magnet, depending on which group of his songs is larger. We feel that the advantages of dominant zooming are more important than this disadvantage. Furthermore, if a user is looking for fast music and finds some artists that he likes, he can still find the other songs (i.e. the very fast songs) from those artists by looking them up in the list.

4.4.2- Zooming design

Zooming in the artist map is geometrical. Zooming in on the region shown in Figure 4.9 is clear: the result is a picture in which the artists with fast music from the 90s are shown. In the picture, two magnet types are used and the contents of the region are clear. This is not always the case, but we would like to tell the user what kind of music is inside the region. Figure 4.10 shows some examples of zooming regions.

In the case that magnets are used, the user can select a region containing:

1. A clearly identifiable kind of artists
2. A collection of multiple kinds of artists
3. Nothing; an empty part of the map

And if no magnets are used, a selected region can contain:

4. A collection of similar artists that cannot easily be named
5. An unidentifiable group of artists
(6. Nothing, an empty part of the map)

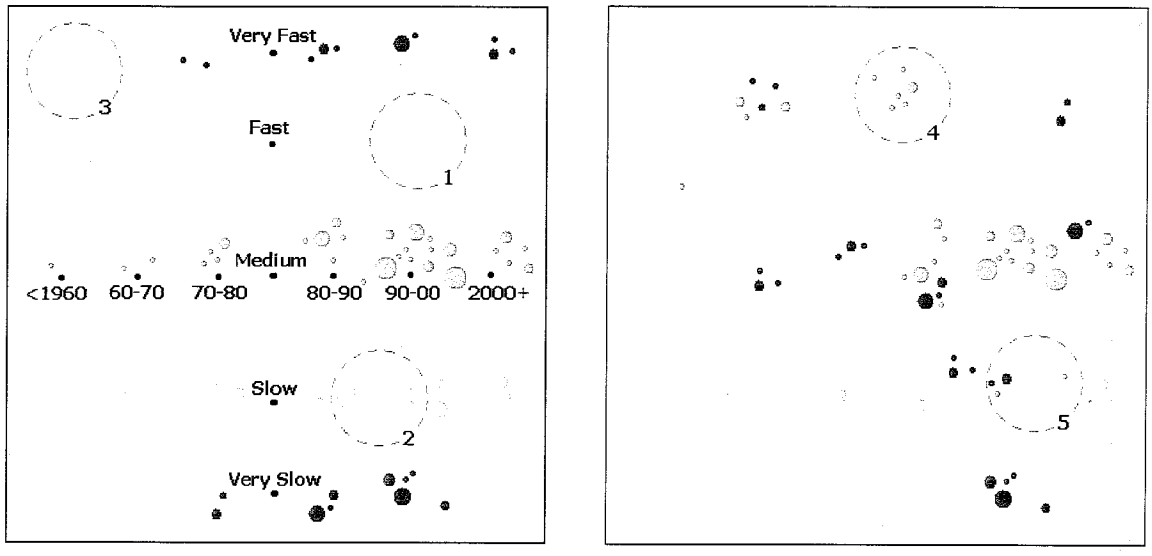


Figure 4.10: Possible zooming regions in an artist map. The user can select a region that contains (1) a clearly identifiable artist kind, (2) a mix of several artist kinds, (3) an empty region, (4) a collection of similar artists that cannot be identified by magnets (labels) because there are none, or (5) an unidentifiable group of artists (not even all similar)

The zooming method itself is geometrical: everything inside the zoom region is enlarged and everything outside of the zooming region is discarded. So it is easy to understand what happens to the picture when zooming is performed in all of the abovementioned cases. However, it is not always clear what kind of music the user is zooming in on, especially in cases 4 and 5. To enhance the ease of use, we want to tell the user what kind of music he has selected with the zoom region, before he actually chooses to zoom. In cases 1, 2 and 3, we can tell the user he is looking at fast music from the 90s for example, or slow music from the 80s and 90s, or nothing at all. In cases 4 and 5, it is harder to tell the user what he is looking at.

To fill this gap, we show the name of the closest artist when a region is selected. Instead of only selecting a region when the map is clicked, the artist closest to the position of the click is also selected and his name is shown on the screen. Also, the interface tells the user which artists are similar to the selected artist. Thus, when a user clicks on the screen he selects a possibly named region (useful for cases 1-3), but also a named artist (useful for cases 4-5). Additionally, names of some[4] other artists are shown on screen. With the artist name information added, users can see what kind of music they are looking at even when they use no attribute magnets, by searching for artist names they know.

The size of the zoom region and the number of levels you can zoom in are arbitrary. In general it seems like a good idea to make parameters like these settable, so the user can change them to suit his needs. On the other hand we want to keep our user interface as clean and simple as possible, so we chose fixed values for both parameters. The size of the region depends on the number of magnet types used and is set such that one entire kind of music nicely fits in the zoom region (i.e. all music from the 90s fits when one magnet type is used, and for two magnet types all medium rock music fits as well). The number of zoom levels is set to three at the moment; after zooming in three times, you cannot zoom further. Instead of a constant number of allowed zoom levels, we previously used this rule: zooming is allowed, as long as more than one artist is visible. The disadvantage of this rule is that if you zoom in on artists that are positioned close together too many times, their size may increase to be larger than the screen.

[4] The set of artist names shown changes while the interface is used. Initially, no artist names are shown and over time the visualized names vary. Some are randomly chosen, but artists similar to the selected one are always named

4.5- Playlist creation

Making playlists on current user interfaces for portable music players can be hard and often takes a long time, because songs have to be selected one by one in long lists. We want our user interface to support fast and easy playlist creation. Based on the artist map framework, the innovative playlist generation method we propose is to *map a playlist path using waypoints*. This playlist creation method is based on the following requirements:

- Little interaction should be required
- The user chooses his level of control
- The playlist contains music that can be expected given the input

The amount of interaction required is of course related to the level of control the user wants to have, but in general creating a playlist of any length should only take a small number of user actions. The playlist will only contain music that can be expected given the input; i.e. no completely random songs are added – only songs that match the criteria specified by the user.

Note that the level of control needs to be within reasonable bounds: this playlist creation method does not allow the user to create a completely hand-picked playlist, let alone speed up the process of creating such a specific playlist. Instead, the highest level of control is such that the user can specify from which artists to play music, in what order, and what kind of songs from these artists should preferably be played. At the lowest level of control, songs are played based on a configuration of attribute values that corresponds to a single specified position in the graph. Examples of both levels of control are given below. First we explain what the user can do to easily create a playlist. Then the inner workings of this playlist creation method are described in general terms, roughly showing how the input is translated into a useful playlist.

4.5.1- Mapping the playlist path

To create a playlist path, given an artist map, the user needs to:

- Specify waypoints
- Specify a number of songs

The waypoints are placed on the artist map and determine the kind of music that will be enqueued in the playlist. All waypoints together form a path that the playlist will follow. Based on the artist map from Figure 4.6, the user could place the four waypoints shown in Figure 4.11 and specify the number of songs to be played along this playlist path – or use the default number. In the picture, four waypoints are placed on a year-tempo artist map. The waypoints are represented by (gray) dots. Using these waypoints and the order in which they were placed, a path is created that represents the playlist. In this example, the playlist starts with slow music from the 60s, and follows with newer and newer songs of about the same tempo. When the playlist reaches music from the 90s, the tempo is increased until the list contains songs with a very fast tempo. Keeping the tempo very high, newer music is added again until the playlist ends with some very recent and very fast songs. Say that the total number of songs in the playlist was set to 24, we will get about eight songs on each of the three parts of the path.

The same way to generate playlists could be applied when there are other types of magnets on the screen, as shown in Figure 4.7 for example. Based on the mood magnets the user could generate a playlist that goes from aggressive to mellow music. Even when no magnets are used at all, a playlist can still be created in the same fashion. In this case, the playlist is composed of songs from the various groups of similar artists that are positioned near the playlist path.

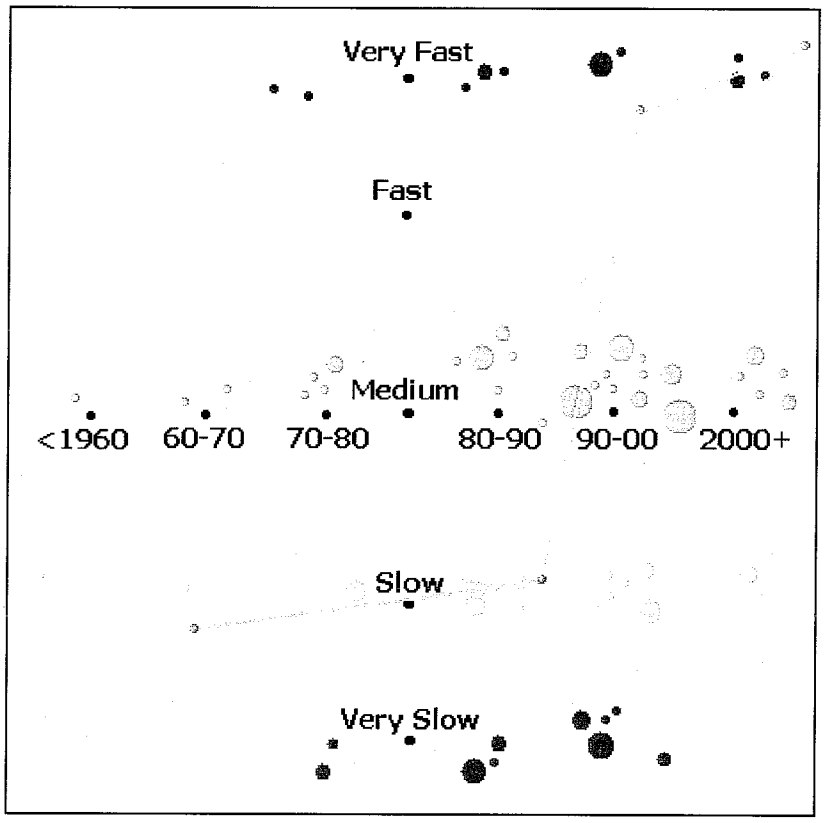


Figure 4.11: An example playlist path, based on the year-tempo artist map from Figure 4.6. The playlist starts with slow music from the 60s, and follows with newer and newer songs of about the same tempo. As soon as the playlist reaches music from the 90s, the tempo is increased until the list contains songs with a very fast tempo. Keeping the tempo very high, newer music is added again until the playlist ends with some very recent and very fast songs

We just described a general example. What about the special cases: the outer bounds of user control? The fastest way to create a playlist 'path', giving up most of your control as a user, is by specifying only one waypoint. The default number of songs for one waypoint is 'unlimited', meaning the device should just keep playing until manually stopped. The songs a user can expect in this case are close to the only waypoint he specified, but move further and further away from it as we choose not to play songs twice. An example is clicking near a mood magnet, say happy, and hearing all happy songs first, but music from all other moods once the happy songs are all played. An optional parameter could specify how far the played items may diverge from the selected point. Note that songs and artists that are not shown in the current view (i.e. after zooming in) are not considered in the creation of the playlist.

On the other extreme, the user may want to have as much control over the playlist path creation as possible. To do so, he can set the number of played songs equal to the number of waypoints placed. If all waypoints are placed near or on an artist, one song from each of these artists is added to the playlist.

4.5.2- Generating the playlist

This section will answer the following question: given an artist map, a number of waypoints and a number of songs to be played, how will the playlist be created? The process of building the playlist is divided into the following steps:

- Generate playlist points
- Compute attribute configurations
- Find matching songs

The playlist points are generated by dividing the path from waypoint to waypoint into smaller segments, depending on the chosen number of songs. For each of the playlist points, an attribute configuration is computed and finally one song is matched to each of these configurations to create the actual playlist. The steps are discussed in more detail in chapter 5. When playing the playlist, its path is shown on the screen, communicating clearly what kind of music the playlist contains. When exporting or sharing the playlist it is possible that not all the songs in the old playlist are available on the new device. However, the system can automatically choose new songs that fit the user requirements because the user did not specify the songs but only gave some criteria.

4.5.3- Playlists and attribute ranges

When creating a playlist, the attribute ranges also present a (minor) problem. If it were true that artists on a certain position in the map only make music that corresponds to that position or region, we could simply add a random song of an artist near the playlist path to the list. Unfortunately, this is not the case. So we need attribute information for all individual songs – this way we can pick a song from an artist that actually does conform to the attributes in this region of the layout. Because of the dominant attraction rule, artists near a magnet are guaranteed to have at least one song that matches the magnet. For the partial attraction scheme this is not necessarily the case.

4.6- Combining search methods

Personal taste is hard to describe, and impossible to predict. To accommodate for the largest possible range of user needs, the developed user interface supports fuzzy as well as specific searches. Sometimes users may know exactly which songs they want to hear, while a moment later they prefer to hear 'anything new' or 'preferably something happy'. In our interface, the artist map is combined with a standard list-based interface. The list supports music searches of the first kind, while the artist map supports the non-specific class of music queries. While the two interface parts are very useful in their own right, combining them in a single user interface extends their use to high above the sum of the parts.

This synergy stems from the fact that people like to narrow down large quantities they cannot handle to manageable proportions. The artist map makes sure that in the search for music that suits your current taste, you never have to bite off more than you can chew: as long as the list is filled with unknown artists, albums, or songs, you simply select a music subset based on much less specific choices in the map. Moreover, using the artist map to find out which artists in your collection make a certain kind of music potentially makes you discover interesting artists again and again. Combining the two search methods helps you rediscover your music collection.

5- Realization

To test the visualization of the similar artist map and its navigation, the designed ideas have been implemented. This chapter explains the technical details of how the envisioned interface was prototyped and revised. The chapter is organized as follows: the first section explains where the similarity metric and attribute information we use originated. Section 5.2 discusses graph-drawing algorithms; the early prototypes of the artist map were used to test various layout algorithms for the positioning of artists. Section 5.3 is about visual enhancements to the computed graph layout. In section 5.4 we discuss the artist map interface that was designed for user tests. Concluding this chapter, section 5.5 gives a description of the implementation work done for the early prototypes and the test interface.

5.1- Attributes and artist similarity

In this section, details about the attribute information used in the artist map are made precise, and our choice of its sources is motivated. Further, we discuss the metric of artist similarity.

5.1.1- Attribute information

The artist map uses various kinds of attribute information: catalogue metadata as well as computable attributes. The catalogue metadata – artist, album, song name and publication year can be obtained from web-services or from information tagged to the files. The tempo attribute is a feature that can be extracted directly from the raw audio data. The mood and genre attributes can be obtained from music classification as described in [23] and [24,25], which is based on feature extraction from the raw audio data. Since the computation of mood, genre and tempo from music data is still in the early stages, it is quite slow and often incorrect. Therefore we used Moodlogic [21] to get attribute information for the songs in our music collection.

However, the speed of computation as well as the quality of the computed attributes is expected to increase significantly in the near future, because of the ongoing research in this field. We expect that in the near future the use of online databases will not be needed anymore to gather all the attribute information that was used in the artist map application. Thus, the classification of music with this interface will not necessarily require an Internet connection: new content can be added to the collection and absorbed by the layout system on the move. Of course this only goes for the computable music attributes such as mood, tempo and genre: for standard metadata such as the artist name and song name we still rely on the completeness of tagged information if we want to refrain from using online services.

5.1.2- Artist similarity

In the artist map, *similar* artists are positioned near each other. To understand what this means, the concept of artist similarity needs to be defined. In this section we discuss the computation of artist similarity, and describe how similarity can be used in the artist map.

Computing similarity

The similarity of artists is based on features extracted from their songs and can be computed for any pair of artists. We used the following method, analogous to that described in [8]. Figure 5.1 shows an overview of the method, by presenting an input-output diagram of all the steps taken.

1. Each song in the collection is processed to obtain feature vectors v_c. The features in one vector form a signature of the sound of the corresponding song. They are based on frequency changes in the song over time. We used features computed as described in [23] which show good discrimination properties, although standard *MFCCs* could also have been employed. Standard MFCCs, Mel-Frequency Cepstral Coefficients, are the short-term spectral-based features widely used for speech recognition software. Logan has shown that MFCCs are also appropriate as a representation for music [26].
2. The feature vectors v_c are *whitened* [27]. Whitening linearly transforms a vector in such a way that the components of the resulting vector are uncorrelated and their variances equal unity; whitened vectors contain independent components.
3. The whitened vectors v'_c are used to train a P×Q Self-Organizing Map (SOM) [28]. The SOM is used to perform vector quantization. Quantization is the procedure of approximating continuous values with discrete values. A vector quantizer maps *k-dimensional* vectors in the vector space R^k into a finite set of vectors $Y = \{y_i: i = 1, 2, ..., N\}$ [29]. If a self-organizing map is used for vector quantization, every element of the map represents one of the vectors y_i . The size of the SOM is arbitrary (in our case the size of the SOM was given by P=Q=8).
4. For each artist A belonging to the collection, a 2d histogram $H_A : [0..P) \times [0..Q) \to \mathbb{R}$ is computed by accumulating the response of the SOM to the whitened feature vectors v'_c of the songs performed by that artist. The vectors y_i were set by training the SOM. By quantization with the trained SOM a histogram is created for each artist. The resulting histogram can be regarded as a probability distribution of artist A's songs in the feature space. The sum over all values in a histogram equals 1, see formula (1)

$$\forall_{A \in artists} \sum_{p=0}^{P-1} \sum_{q=0}^{Q-1} H_A(p,q) = 1 \qquad (1)$$

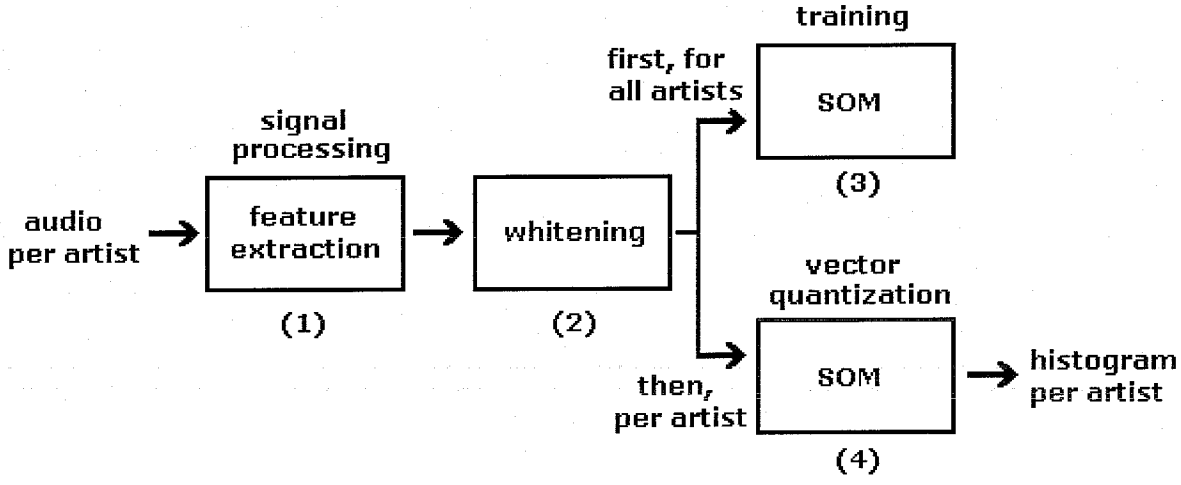


Figure 5.1: Input / output diagram for the process of creating a histogram that represents the sound of an artist. For each artist, features are extracted from their songs (1), the feature vectors are whitened (2) and used to train a self-organizing map (3). This SOM is used to perform vector quantization (4) and thus create a histogram for each artist, representing its songs in the feature space. The histograms are used to compare the artists: the artists are similar (sound similar) if their histograms are alike

In figure 5.2 the histograms computed for two artists are shown: Abba and Pearl Jam. From the figure we can infer for the music in the available collection that Pearl Jam has a lot more variety in its music than Abba.

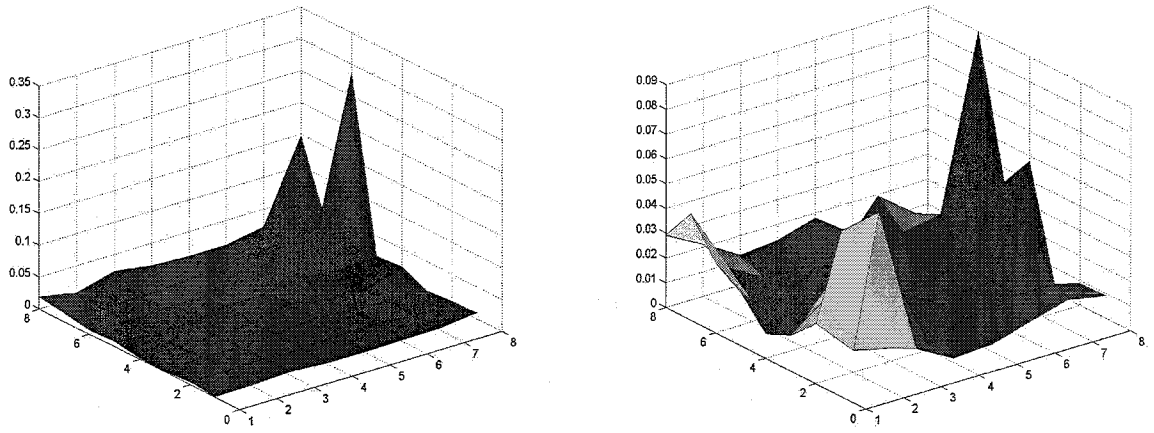


Figure 5.2: Two example histograms. The Abba histogram is shown on the left, and the Pearl Jam histogram is on the right. It can be seen that in the corresponding music collection, Pearl Jam has more song variety than Abba, and the music of the two artists sounds differently

Although it is hard to name the kind of music associated with a certain point or range in the histogram, we can say that two artists are similar if their histograms are alike. Measuring the amount of overlap in their histograms, the similarity Sim(A, B) between two artists A and B is defined as follows (taken from [30]):

$$Sim(A,B) = \sum_{p=0}^{P-1}\sum_{q=0}^{Q-1} \min(H_A(p,q), H_B(p,q)). \qquad (2)$$

This similarity metric is mainly chosen because it maps similarity to a number between zero and one. Further, it is intuitive because the similarity of two artists equals the 'percentage' for which their sound matches – since *sound* is the information contained in the histogram. The similarity only equals one if the two artists have completely identical histograms. It is very unlikely that two different artists have this maximum similarity; artists only sound totally equal to themselves.

Derived similarity metrics

Although formula (2) gives a clear definition of artist similarity, the artist map needs derived information to compute the positioning of artists[5], since instead of a continuous similarity value that signifies the proximity of two artists, some positioning algorithms need a *distance* between the artists and some need a *binary* relation (i.e. two artists are either similar or dissimilar). To be able to use such algorithms, we define a binary similarity relation and a similarity distance.

Using a threshold **t**, $0 \leq \mathbf{t} \leq 1$, we say that artists are similar if their similarity value is at least **t**.

A **is similar to** B $\equiv$ Sim(A,B) $\geq$ **t**

This similarity relation is both *reflective* and *symmetric*, but not *transitive*. Reflective means that all artists are similar to themselves: A is similar to A because Sim(A,A) = 1, and $1 \geq \mathbf{t}$. The relation is also symmetric: Sim(A,B) = Sim(B,A). The fact that the similarity relation is not transitive is less straightforward, and is proven in appendix C.

The similarity distance Dist(A,B) is defined as follows:

Dist(A,B) = 1 – Sim(A,B)

[5] The positioning of artists is computed by *graph layout algorithms*. For more information about these algorithms and the graph-representation of the artist map, refer to section 5.2

An advantage of this distance definition when compared to e.g. the Manhattan distance is that it maps to the small range of 0 to 1. Still, it is an actual distance, as we will prove below. There are four properties of a distance measure ρ:

1. $\rho(x,y) \geq 0$

2. $\rho(x,y) = 0 \Leftrightarrow x = y$

3. $\rho(x,y) = \rho(y,x)$

4. $\rho(x,z) \leq \rho(x,y) + \rho(y,z)$

Since similarity distance has a range from 0 to 1 and Sim(A,B) is only equal to 1 if A and B are equal, property 1 and 2 are trivial. Sim(A,B) = Sim(B,A) proves property 3. Property 4, the triangular inequality, also holds for similarity distance: Dist(A,C) ≤ Dist(A,B)+Dist(B,C), as we prove below for all artists A, B and C:

$$Dist(A,C) \leq Dist(A,B) + Dist(B,C)$$

$\equiv$

$$1 - Sim(A,C) \leq 1 - Sim(A,B) + (1 - Sim(B,C))$$

$\equiv$

$$Sim(A,B) + Sim(B,C) \leq 1 + Sim(A,C)$$

$\equiv$

$$\sum_{p,q} \min(H_A(p,q), H_B(p,q)) + \sum_{p,q} \min(H_B(p,q), H_C(p,q)) \leq 1 + \sum_{p,q} \min(H_A(p,q), H_C(p,q))$$

$\Leftarrow$ { min(a,b) + min(b,c) ≤ b + min(a,c) }

$$\sum_{p,q} H_B(p,q) + \sum_{p,q} \min(H_A(p,q), H_C(p,q)) \leq 1 + \sum_{p,q} \min(H_A(p,q), H_C(p,q))$$

$\equiv$

$$\sum_{p,q} H_B(p,q) \leq 1$$

$\equiv$

true.

5.2- Visualizing layout algorithms

The artist map is implemented as a complete, edge-labeled graph G = (V, E), in which the vertices are artists and the edges represent similarity distance relations on the artists. The graph is complete since for each pair of artists the similarity distance can be computed. The *drawing* of graph G is a function p: V $\rightarrow \mathbb{R}^2$, that maps the vertices of the graph to positions in two dimensions. The whole of vertex or artist positions is also called the *layout* of the graph. In our visual representations of graph drawings we choose not to include edges, because they clutter the visualization. The drawing of an artist graph is important because the positioning of artists in the layout is crucial to the effectiveness and usability of our visualization. In this section we discuss how similarity information is used in the computation of a layout, and how the layout is computed. Various examples of layout algorithms are given, and we explain which algorithm we ultimately used to create a layout that presents a clear overview of a music collection.

5.2.1- Similarity and layout algorithms

Graph G seems to be perfectly fit to create a nice graph drawing: the foremost requirement is to linearly map the similarity distance from graph G to a geometrical distance in the visualized layout. However, using the similarity distance as input for graph-drawing algorithms (by linearly relating forces in the algorithm to the similarity distance between artists) proved to be less useful than we anticipated. This may be caused by the fact that most of the used layout algorithms were designed with binary relations in mind (i.e. they use unlabeled edges). Instead of using the similarity distance, we construct a new graph based on our binary similarity relation.

In the new graph, G' = (V', E'), two artists A and B are connected by an edge if they are similar, i.e. if Sim(A,B) ≥ **t**. We want the drawing of this graph to be *nice*. A nice drawing exhibits the following properties:

i. Similar artists are placed close together
ii. Clusters of similar artists can be easily identified
iii. Artists are distinguishable

To create such a layout, we used and adapted force-directed graph-drawing algorithms to be able to satisfy these properties. Such algorithms use a physical analogy to compute the layout of a graph, where the graph is seen as a system of bodies with forces acting between them. A force-directed graph-drawing algorithm can be regarded as an optimization process that seeks a configuration of the bodies with locally minimal energy. Such an equilibrium configuration, in which the sum of the forces on each body is zero, is expected to correspond to a *nice* drawing – although it depends on the used force-model. In the next sections we discuss how we arrived at our final graph-drawing algorithm, starting with the Spring Embedder algorithm.

5.2.2- Spring embedder

The Spring Embedder method by Eades [31] is among the first applications of force-directed methods on graph drawing and evolved from the VLSI technique of force-directed placement. In the spring embedder algorithm, the physical analogy used is as follows: vertices are replaced by charged particles that repel each other, and edges are replaced by springs that connect the particles. Figure 5.3 gives a conceptual idea of how the spring embedder algorithm works. Given a graph and its initial drawing in which vertices have a random position, we assign attraction and repulsion forces, let the system go to find a low energy state, and end up with a drawing of the graph that looks clearer. The configuration found in a low energy state depends on the exact force-model used and the initial positions of the vertices.

General pseudo-code of force-directed graph drawing algorithms: (unit(x) computes a unit vector by dividing vector x by its length, and returns (0,0) if the length of x, written as |x|, is 0)

```
G = (V,E)
for all v in V : p_v = random distinct position  // all initial positions are different
while (! stop_criterium) { // stop_criterium could be max number of iterations or minimal total displacement
        // calculate repulsion forces
        for v in V {
                disp_v = 0; // disp is the cumulative displacement vector for a vertex
                for all u!=v in V {
                        Δ= p_v – p_u;
                        disp_v += unit(Δ) * repulsion( | Δ | );
                }
        }                                                                              (3)
        // calculate attraction forces
        for all edges {u,v} in E {
                Δ= p_v – p_u;
                disp_u += unit(Δ)  * attraction( | Δ | );
                disp_v -=  unit(Δ)  * attraction( | Δ | );
        }
        // apply the displacements, possibly limited by a temperature
        for v in V {
                p_v += unit(disp_v) * min(|disp_v|, temperature);
        }
        decrease temperature;
}
```

Notice the *temperature* used in the layout algorithm. The temperature limits the maximum displacement of a vertex per iteration (the idea comes from simulated annealing, [32]). Starting with a high temperature and slowly cooling it down can improve the speed of convergence to an aesthetically pleasing layout. The temperature can also be used to determine the number of iterations. What discriminates one force-directed graph-drawing algorithm from another is the used force-model. The force-model encodes the aesthetic criteria that a layout created with the algorithm should satisfy. In the pseudo-code above the force-model resides in the attraction and repulsion functions. For the layout algorithms we discuss, these force functions will be listed. In general, the force F exerted on a vertex v can be expressed as follows [15]: (where $\Delta_{v,u} = p_v - p_u$)

$$F(v) = \sum_{\{u,v\} \in edges} attraction(|\Delta_{v,u}|) unit(\Delta_{v,u}) + \sum_{u \in vertices, u \neq v} repulsion(|\Delta_{v,u}|) unit(\Delta_{v,u}) \qquad (4)$$

Fruchterman and Reingold proposed a modified version of the spring embedder algorithm that more closely models the properties we are interested in than the original model by Eades. Their force-model is implemented as follows [33]:

attraction(δ) = δ^2 / l

repulsion(δ) = $l^2 / (\delta + (l^2 / \text{MaxRep}))$

constant l specifies the optimum distance between two vertices. MaxRep denotes the maximum repulsion force and has been added to the repulsion function to prevent a singularity for $\delta = 0$

So the attraction force exerted on neighboring vertices is related to the distance between them squared, and the repulsion force on two vertices is inversely related to their distance. Notice that the strength of the connection between two vertices is not mentioned. Parameter l is a constant, so only the distance between two vertices and this desired distance constant are used. This is an example of a graph-layout algorithm that uses a *binary* relation.

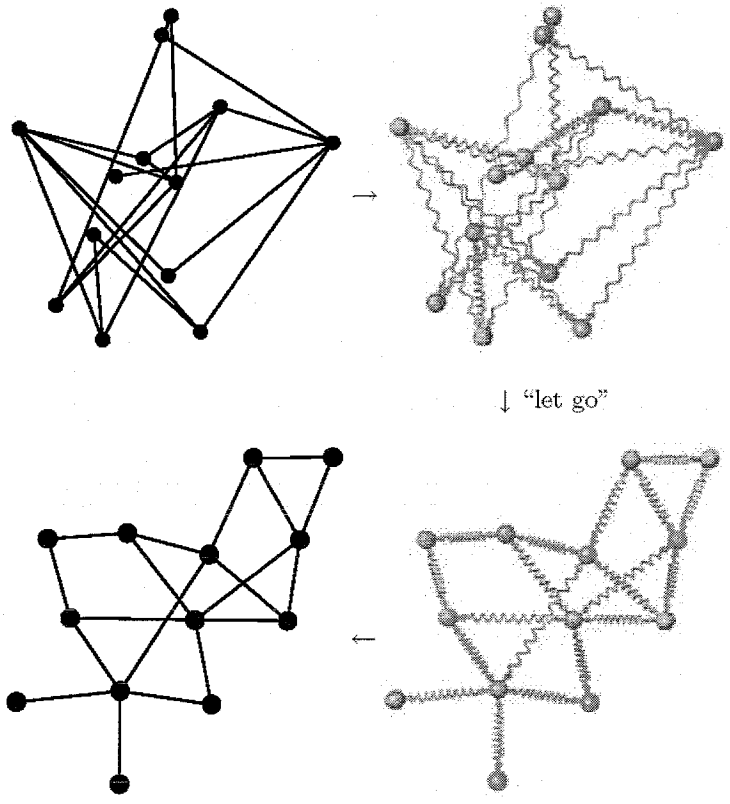


Figure 5.3: Spring-Embedder algorithm example, after [34]. The final layout (bottom left) looks clearer than the random initial drawing (top left)

Applying the simple Fruchterman-Reingold model directly on an artist test-graph that has four clusters and four sub-clusters for each cluster, results in a graph drawing that is not *nice* (see Figure 5.4). Closer investigation shows that there are two problems:

1. The vertices are too evenly spread; the grouping (or *clustering*) is not obvious
2. The position of clusters is not the same for subsequent runs of the algorithm

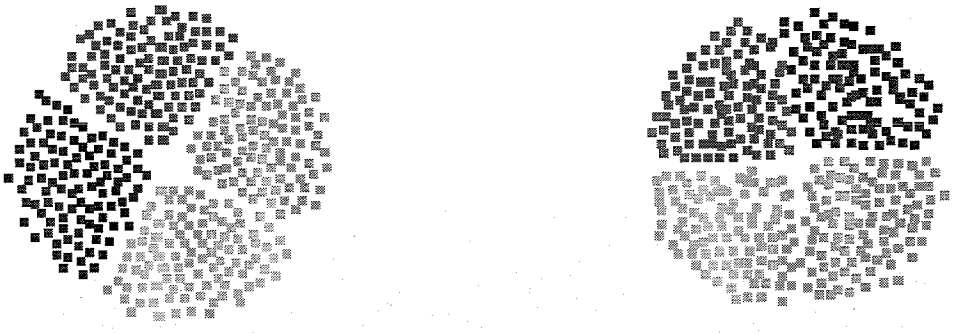


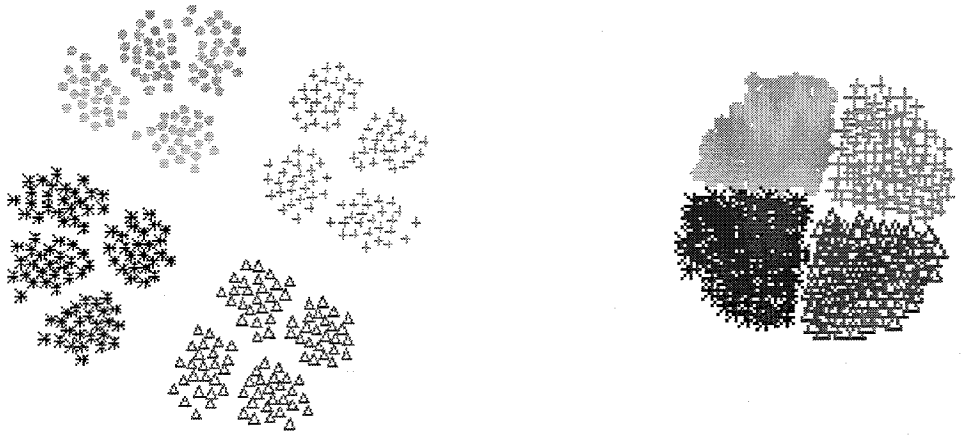
Figure 5.4: The layouts resulting from two different runs of the Fruchterman-Reingold algorithm on the same test-graph. Although four groups are slightly separated in both layouts, the position of one such a group is not the same in subsequent runs (i.e. the darker items are on the left in the left picture and on the top in the right picture). Moreover, each group has four sub-groups that are not visually separated at all. Therefore, another force-model is needed

5.2.3- LinLog

With the Fruchterman Reingold (FR) model, clusters are not obvious in the resulting layout. In chapter 3 we discussed the use of *sharpening* to improve such layouts: if similar artists are positioned close together by the original graph layout algorithm, these groups of similar artists are accentuated by sharpening. A disadvantage of sharpening as a post-processing step is that the quality of the resulting layout highly depends on the initially used layout algorithm. If similar artists are not close in the layout before sharpening, the sharpening step may suggest artist relations that do not really exist. Since groups of artists are accentuated after sharpening, you expect the close artists to be similar more than you would in the initial layout. To make sure that similar artists form clusters in the final layout, we use an algorithm that accentuates the clusters as an integral part of the layout forming process. A. Noack has developed such a graph layout algorithm and compared it to several other layout algorithms [35]. He showed that his algorithm produces a *clustering*.

> **Clustering** – A *clustering* is a graph layout in which clusters (groups) of neighboring vertices are visually separable, even if the edges are not drawn. In a clustering, the distance of two cohesive sub-graphs is inversely proportional to their coupling. Noack makes this property more precise in [35].

Therefore, to improve the clustering properties of the layout, we changed the force model based on [35], in which an energy model is introduced that produces a more obvious clustering: the *LinLog* model. We have tested and compared the results of *LinLog* to those of FR-based models using a test collection of graphs obtained from [35] and extended for our purposes. Some of our results are shown in Figures 5.5 and 5.6. Although the graphs used in Figure 5.5 a) and 5.5 b) are exactly the same, the LinLog drawing a) shows the clusters much clearer than drawing b) that was produced by a force-directed method based on FR.



a) b)

Figure 5.5: Results of clustering tests: a) LinLog graph layout method, versus b) graph layout method based on Fruchterman-Reingold. The LinLog method results in a layout in which (sub-) clusters are much clearer. White space helps to make the clusters and sub-clusters stand out

Given a graph *G(V,E)* where *V* is a set of nodes and *E* a set of edges, $p: V \rightarrow \mathbb{R}^2$ is the two-dimensional drawing of *G*. The *LinLog* energy model $U_{LinLog}(p)$ is defined as:

$$U_{LinLog}(p) = \sum_{\{u,v\} \in E} |p_u - p_v| - \sum_{\{u,v\} \in V^2, u \neq v} \ln(|p_u - p_v|). \quad (5)$$

In this formula, p_u and p_v are the positions in the drawing of nodes *u* and v respectively. The first term represents attraction between connected vertices, while the second term represents repulsion – which separates clusters of similar artists and avoids overlapping vertices.

From this LinLog energy model a force model can be derived (see [36]), which we implemented as shown below. Note that although the LinLog model produces a better clustering, the positions of clusters are still different in various runs of the algorithm.

attraction(δ) = k

repulsion(δ) = r / (δ + (r / MaxRep))

constant k specifies the stiffness of the springs, r specifies the repulsive (electrical) force of the bodies, and MaxRep denotes the maximum repulsion force. MaxRep was added to the repulsion function to prevent a singularity for δ = 0

In the LinLog model, the attraction force is constant, and the repulsive force is inversely related to the distance. Again the parameters *k* and *r* are constants and do not depend on the actual artists involved. Therefore, the continuous similarity distance is not used.

5.2.4- Extended LinLog

The second problem we identified is not always a problem for graph drawings in general: as long as the clusters itself are obvious, the positions of the clusters often do not matter. However in the case of music visualization, we desire geometrical dimensions with a clear meaning. For various runs of the graph layout algorithm, clusters of artists with the same 'kind' of music should not change position much.

To obtain this result, the LinLog algorithm was extended with attribute magnets for each of the attribute types: *mood*, *genre*, *year* and *tempo*. In our implementation, dummy attractors represent these magnets (note that the magnets are not part of the set of vertices *V* itself).

In the extended LinLog model, the energy $U_{MLinLog}(p)$ of a drawing $p: V \cup M \rightarrow \mathbb{R}^2$ is defined as:

$$U_{MLinLog}(p) = U_{LinLog}(p) + \sum_{\{v,m\} \in V \times M} affinity(v,m) \frac{k}{2} (|p_v - p_m| - l)^2 \qquad (6)$$

The second term represents the attraction of the artists to the magnets, which depends on their distance and an *affinity* scale. *M* is the set of magnets. The corresponding force function, which we call *magnetAttraction*, is proportional to the distance δ between an artist *v* and a magnet *m* and is defined as follows (before scaling according to the affinity function):

magnetAttraction (δ) = ($\delta - l$) * *k*
constants *l* and *k* represent the preferred spring length and spring stiffness respectively

So the force exerted on vertex v by magnet m equals *affinity(v,m)* magnetAttraction*($|p_v - p_m|$). The *affinity:* $V \times M \rightarrow [0,1]$ computes the affinity of artist *v* with the music represented by magnet *m*. For example, artists who perform only happy music have affinity equal to 1 for the *happy* magnet, while the same artists have affinity equal to 0 for all other moods. For each artist the sum of affinities for any given magnet type equals 1:

$$\forall_{v \in V} \sum_{m \in M_M} affinity(v,m) = \sum_{m \in M_G} affinity(v,m) = \sum_{m \in M_Y} affinity(v,m) = \sum_{m \in M_T} affinity(v,m) = 1$$

where M_M, M_G, M_Y, M_T are the four different magnet groups (mood, genre, year, tempo).

In Figure 5.6, test results are shown in which extended LinLog and the method based on Fruchterman-Reingold are compared when four mood magnets and four genre magnets are used. Again, the resulting layout is *nicer* for the *LinLog* method, shown in Figure 5.6 a).

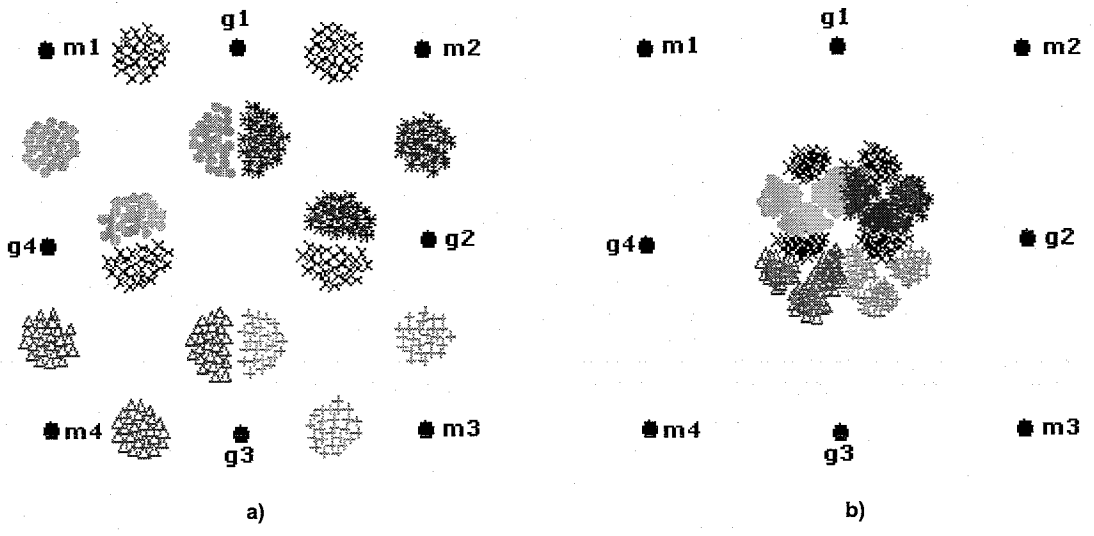


Figure 5.6: Results of clustering tests with attribute magnets. Four genre magnets and four mood magnets are used. a) shows the result of the extended LinLog method, which is compared to the Fruchterman-Reingold method in b). Clusters are clearer when the extended LinLog method is used. Besides guaranteeing that the same kind of music always ends up at roughly the same position, magnets are labels that show what kind of music ended up where. I.e. the four clusters at the bottom left of a) contain artists that mainly make music of mood m4, and in this group the left sub-cluster contains artists that make music of genre g4

For the extended LinLog algorithm, the attraction and repulsion functions do not have to be changed. Instead, an extra loop is added to the algorithm, which attracts vertices to the used magnets using the *magnetAttraction* and *affinity* functions. An added benefit of using magnets is that the computation of the layout can be faster: only a few iterations are needed before the layout reaches an acceptable state, especially if the attraction to magnets is stronger than the attraction to similar artists. This is caused by the fact that the layout algorithm does not move the magnets, so the clustering enforced by magnets can be reached in a small number of steps.

The *affinity* is particularly useful when partial attraction is used. In our case, a magnet m only attracts an artist v if the affinity of v with magnet m is higher than the affinity of v with any other magnet of the same kind (*affinity*(v, m) ≥ *affinity*(v, n), for all magnets n of the same type).

5.3- Visualization enhancements

Using the discussed graph layout algorithms, we created a *nice* drawing of a graph in which clusters of similar artists are clearly separable. However, the visualization of the graph can be improved significantly by adding contextual information. Apart from extending and improving the visualization, such information can also play a role in rating the quality of the layouts created by the algorithms. Coloring is used to visualize the value of a particular attribute type for all artists. Vertices are visualized as circles instead of dots: the size of a circle represents the number of songs available for the corresponding artist. To help satisfy property (iii), "vertices should be distinguishable," distance computations now incorporate the radii of vertices to reduce overlap[6]. The interface shows which magnet types are used and what coloring and attraction rules are in effect. *Partial attraction* and *range coloring* are based on the partial attraction idea from chapter 4, while *most dominant element attraction* and *most dominant element coloring* are based on dominant attraction. Finally, the interface can be used to change all settings and to play with the visualization technique. Figures 5.7, 5.8 and 5.9 show screenshots of the interface, in which different coloring rules and magnet types are visualized.

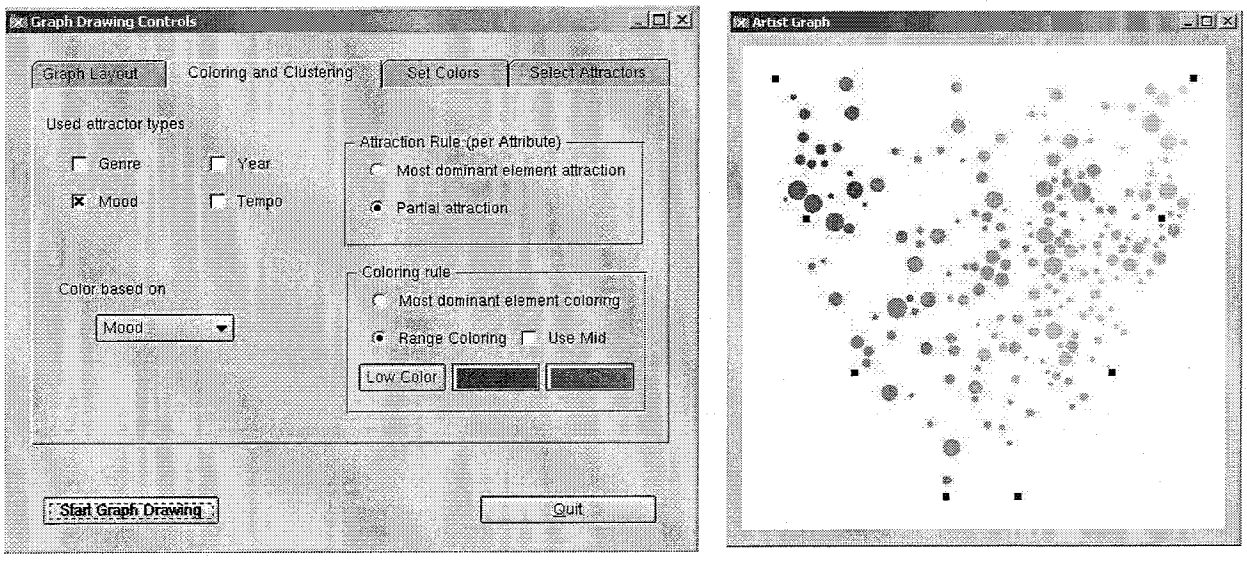


Figure 5.7: Screenshot of the prototype. Context and control are external to the visualization. The size of a circle conveys how many songs there are available for the corresponding artist. Mood magnets are used, with the *partial attraction* rule. Coloring is also done on the mood attribute, following the *range coloring* rule. An artist with 50% happy and 50% sad songs is attracted to both of these magnets with equal strength, and its color is based on both moods.

[6] With the radius of a vertex or artist, we refer to the radius of the circle that visually represents this artist. Distance between these circles is not the distance between the midpoints, but the distance between their boundaries

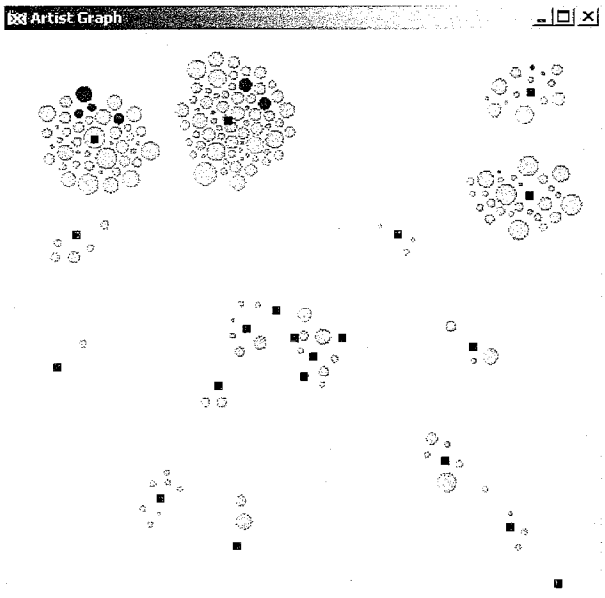


Figure 5.8: An example artist map layout with genre magnets for *dominant* attraction, and *most dominant element* tempo coloring

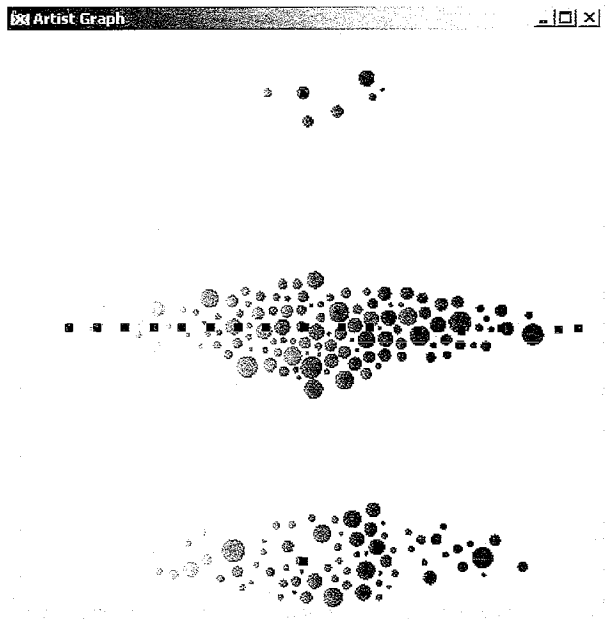


Figure 5.9: An example artist map layout with tempo and year magnets for *dominant* attraction, and *most dominant element* year coloring

Coloring and positioning give useful feedback to the user, and can be used to determine the quality of a layout. However, this way of providing context is hardly good enough for an actual user interface. The most important reason is that the contextual information is external to the visualization. So, we need to develop a more compact and easier to use interface for user tests.

5.4- The user interface

This section describes the realization of the navigation and interaction features of the artist map interface: user-defined context, zooming, playlist creation, and search capabilities. Additional usability requirements were defined to create an interface that better facilitates user tests.

Besides meeting the original requirements from chapter 2, the artist map interface should

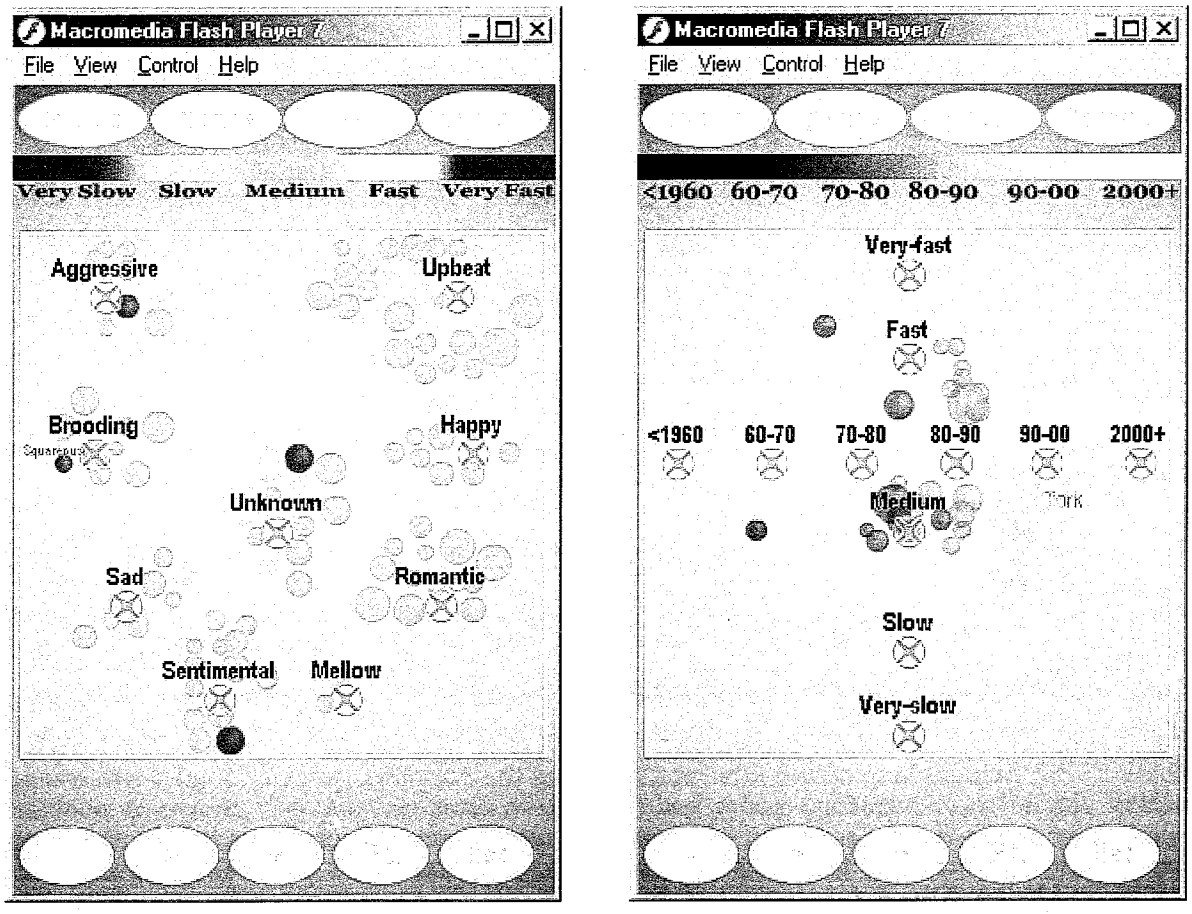
- be clean – less options means easier to use
- be clear – context should be obvious and integrated with the visualization
- be testable on a touch-screen – small, and all input is clicking / dragging
- look nice – people are not going to use it if it looks like a scientific tool

5.4.1- User defined context

Pictures of the new interface are shown in Figures 5.10 and 5.11. The interface contains the artist map as well as the hierarchical list, to support specific as well as non-specific music searches. Contextual information is integrated in the map visualization by showing a legend and labeled magnets. Labels in the visualization provide additional information. The labels describe the selected artist and region if the user clicks in the map, and give information about random artists if the user is not active. There are fewer options in the new interface: all parameter settings are removed. Moving magnets and changing the coloring or positioning is easy and possible just by clicking and dragging. Except for looking better and being ready for testing, this interface allows for the implementation of new features, described in the rest of this chapter.

Figure 5.10: Screenshots of the interface, applied to a music collection of 111 artists and 2248 songs, showing navigation based on a hierarchical list, taken from [37]. On the left, seven genres are shown of which popular is selected. On the right we see the artists on the device belonging to this particular genre: there are 17 artists with popular music available.

The new interface has fewer options, which makes it cleaner and easier to use than the early prototypes. Coloring, button highlighting, labels, and extra textual information about the current view provide contextual information, which improves the clarity of the interface. The artist map interface is designed to be used on a small screen with only simple interaction options (i.e. clicking and dragging), which allows us to perform user-tests on a touch-screen.



a) b)

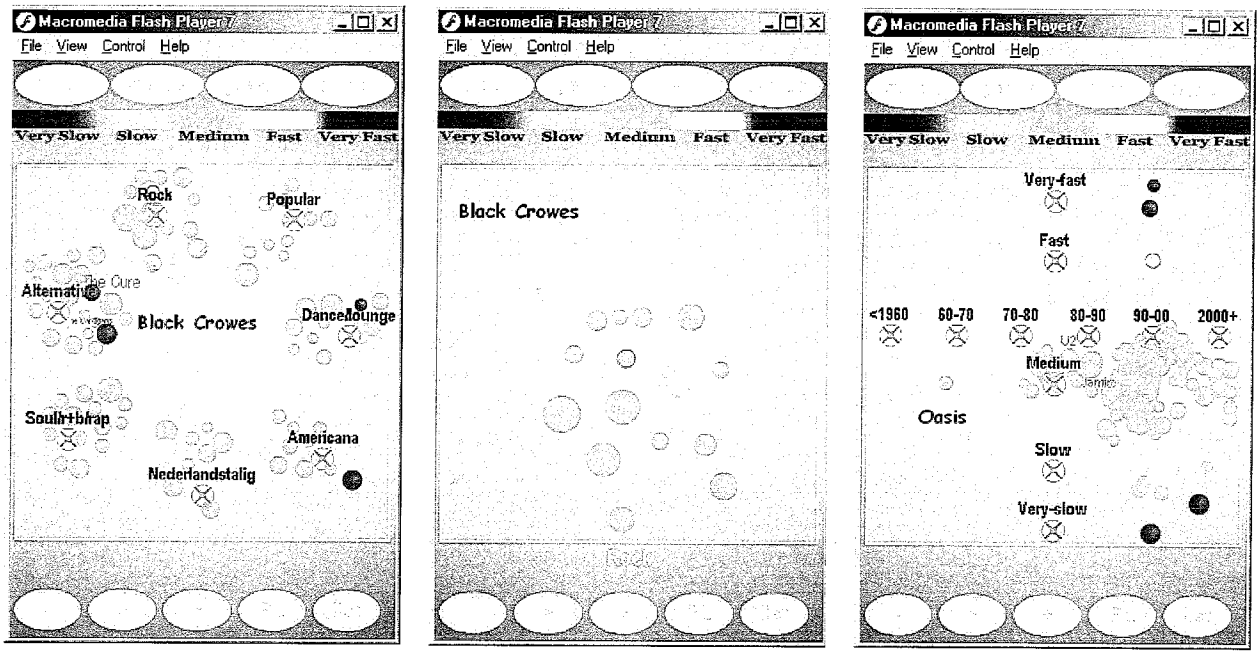
Figure 5.11: Screenshots of the interface, applied to a music collection of 111 artists and 2248 songs, showing a) a *mood-map* in which clustering is based on *mood* (spreading the magnets to make efficient use of screen-space) and coloring is done on *tempo,* and b) a *year-tempo* map where clustering is based on *year of release* along the horizontal direction and *tempo* along the vertical direction, and coloring is done on *year*

5.4.2- Zooming

Because we chose to use dominant attraction only in the artist map, zooming is implemented as a straightforward geometrical zoom. The process of a zooming action is simple:

1. The user selects a position on screen to zoom in on
2. The zoom-region corresponding to this position is visualized
3. The user clicks the 'zoom in' button
4. Everything within the region is enlarged, and everything outside it is discarded

The zooming process is depicted in Figure 5.12.



a) b) c)

Figure 5.12: Screenshots of the interface, showing the zooming process. In a) a *genre-map* in which clustering is based on *genre* and coloring is done on *tempo* is shown. A position is clicked for which Black Crowes is the nearest artist. The region corresponding to the selected position is visualized by the white dashed circle. In b) the result of clicking the zoom-in (+) button is shown. The user has zoomed in on Rock music. Black Crowes is still selected. In c) the zoom-region is depicted in a *year-tempo* map. The zoom-region is smaller in the case of 2 magnets because it should be possible to select fast music from the 90s without including medium or 80s music

5.4.3- Playlist creation

Unfortunately, playlist creation has not been realized because we lacked the time to do so. In chapter 7, playlist creation is mentioned as a recommendation for the future.

5.4.4- Search capabilities

The integration of specific and non-specific searches has an important advantage: the size of the collection browsed in the list interface can be reduced by non-specific selections in the artist map. Zooming and playlist creation in the map are visual actions that influence the items shown in the list. After zooming in on a certain kind of music on the map, say fast and happy, the list is updated to contain only the artists that are still visible. Furthermore, songs that match the last zooming action may be highlighted in the list. I.e. after zooming in on happy music, an artist with 60% happy and 40% sad songs is visible on the map, and also remains in the list. The happy songs from this artist can be highlighted in the list as a result of the zooming action on the map.

5.5- Implementation

In this section we discuss the actual implementation of the layout algorithm prototypes and the artist map interface. We describe what languages and development environments were used, explain the architecture of the artist map interface, and shortly state the coding effort involved.

5.5.1- Layout algorithm prototypes

The implementation of the artist map started with prototypes to test the layout algorithms as well as the initial visualization ideas. Because of the initial aim to support integration with a larger general-purpose Qt-based interface for portable music players developed in the PiC project at Philips Research, the Qt [38] toolkit was used for prototypes. Qt is a cross-platform C++ application development framework. Rapid prototyping of the artist map was possible by using Qt Designer (shown in Figure 5.13), a visual user-interface design tool and code editor for Qt.

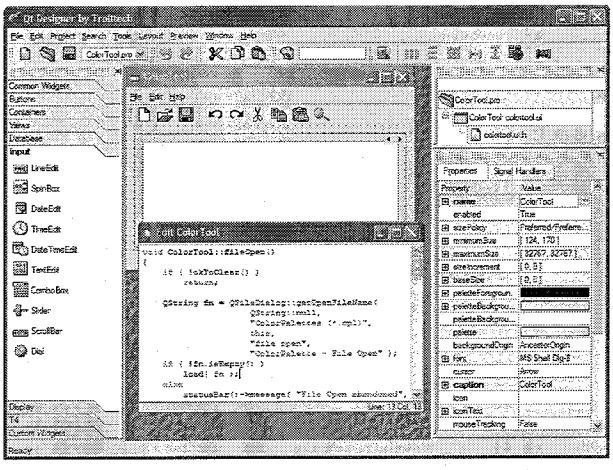


Figure 5.13: A screenshot of Qt Designer, a visual user-interface design tool and code editor for Qt

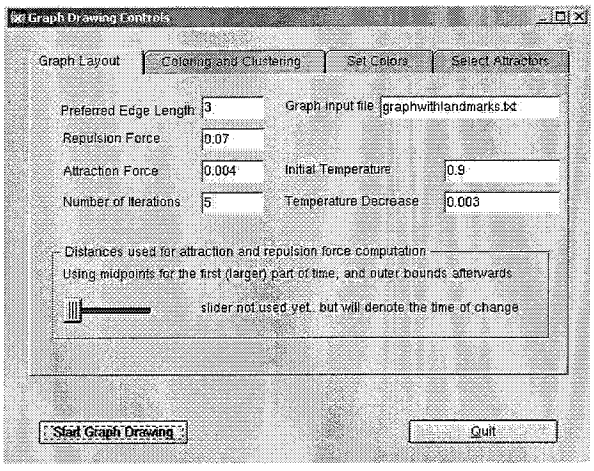


Figure 5.14: The first prototype of the artist map user-interface, created with Qt Designer

Qt Designer enables you to create a functional user interface, such as the one shown in Figure 5.14 in a matter of minutes. OpenGL was used in the early prototypes to render the visualization of the artist map itself in a separate window. Although this setup worked well for the prototypes, where a lot of parameter tweaking had to be done and the amount of functionality needed in the interface was large, the user interface for testing had other important aspects and requirements.

Creating a new interface where everything is nicely integrated and custom-designed can be a very time consuming task when using the prototypes setup of C++, OpenGL and Qt. To shorten the development time, we wanted to use Macromedia Flash – an application for developing rich content, user interfaces, and web applications [39]. However, to keep integration with the PiC project possible, continuing development in the Qt framework instead was an important alternative. The tradeoff we thus encountered was, in short, as follows: *usability over integration or integration over usability*? Either we use all the tools necessary to create a more complete interface that looks like it can be used directly on a portable music player and that users can actually test on a touch-screen, or we work toward integration with a general-purpose interface while risking the fact that the developed interface will not be tested in the near future.

We decided that finding out whether the developed ideas are actually useful is more important than working toward integration with a larger code-base. If the interface is found to be useful, the early prototypes can always be improved at a later time, based on the new Flash interface.

5.5.2- Architecture of the user interface

Various user interface projects at Philips Research are based on Mma; a multi-modal architectural framework that is described in detail in [40]. Although the artist map is not designed as a multi-modal interface, the Mma framework has a lot to offer for its implementation, as it was designed to enable fast prototyping of graphical user interfaces. Mma has an application independent component that is useful for our purposes: the dialogue engine. The dialogue engine is a state machine (defined in an XML file), in which each state corresponds to a state of the interface. As such, it determines the interaction path of the application.

Mma is implemented in Java and is based on clusters and workers. A cluster is the structure that enables workers to communicate with each other, using messages. Each worker is a Java thread and has a companion XML file that describes the messages this worker has subscribed to. Worker threads have a fixed structure as they are all derived from the same abstract super-class. To send messages to another worker, they use the post method – the cluster makes sure that the message reaches the recipient. The Mma framework can easily be integrated with Flash, using XML sockets to make communication between the Flash interface (front-end) and a Java worker (back-end) possible [40]. This way, Flash can be used to create a nice visual interface, while the functionality of the interface is not bound by limitations of the interpreted ActionScript language (the integrated scripting language of Macromedia Flash). The structure of clusters and workers is depicted in Figure 5.15 below.

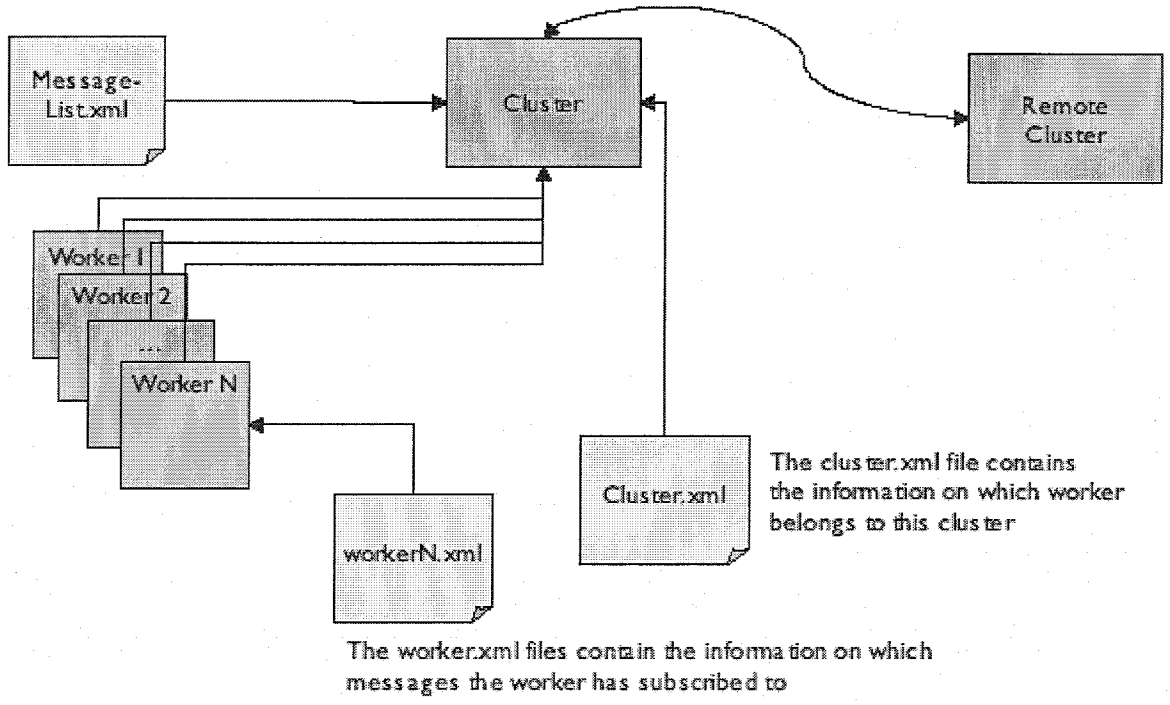


Figure 5.15: Implementation of Mma: relationship between Workers and Cluster, taken from [40]. In our implementation, the functionality of the artist map visualization resides in one Worker, while e.g. the Flash front-end, the list-based interface and the music playback functionality reside in other Workers

Since the Mma framework has already been implemented and successfully tested in many projects at Philips Research, it makes sense to use it for the artist map as well. An initial list-based interface that uses Flash for the front-end and Java for the functionality was available from previous work [37], and the new artist map implementation was based on this interface.

5.5.3- Coding effort

Implementing the layout algorithms, the visualization techniques and the artist map interface has been an interesting programming effort. Different languages and different styles have been adopted for the development of the various visualized layout algorithms and interfaces.

Initially, C++, the Qt framework, and a Linux environment were used to make sure integration with the PiC project would be possible. OpenGL was used to visualize the layouts. For the implementation of the early interfaces, a rapid prototyping approach was adopted. The interface was secondary to the implementation and testing of the layout algorithms. The final prototype to test graph layout algorithms is a program consisting of ~2000 lines of C++/Qt/OpenGL code.

After choosing the layout algorithm to be used, the actual interface still had to be developed. We decided that having a fun, easily understandable and useful interface was more important than integration with the PiC project. This led us take a completely different approach for the testable artist map interface: the Mma framework developed at the Media Interaction group of Philips Research was used to create a Java/Flash based implementation of the artist map. The artist map implementation contains ~2000 lines of Java code and ~800 lines of ActionScript.

To measure the enjoyability, ease of use and usability of the artist map interface, we performed user tests. Both user tests are discussed in the next chapter.

6- User evaluation

This chapter describes the evaluation of the user interface we developed. Two user tests were performed: an informal, qualitative test during development, and a more thorough user test after development of the final prototype. We now discuss both tests and their results.

6.1- Qualitative usability test

The first user test was performed during the development of the artist map interface, to find out what the strong and weak points were so far. This test was performed with 3 goals in mind:

- To find usability problems
- To find out whether people like to use the interface (is it fun, is it easy?)
- To get suggestions for the further development of the interface

In this section we describe the setup of the qualitative usability test, and show the results gained from it. After this test, the interface was revised and tested more thoroughly.

6.1.1- Setup

Seven persons (5 male, 2 female) participated in the qualitative usability test. The test was performed on a desktop pc on which the interface was running. Although the interface was designed to be used on a touch-screen, participants used a mouse to interact with the system in this first test. Also, no actual music could be played. During a 1-hour test session, the participants tried out the interface and were asked a number of questions.

Letting participants try the interface before discussing it with them (throwing them in the deep) was useful to quickly find usability problems. After having played with the interface, the participants answered a number of questions related to the ease of use, design, enjoyability and effectiveness of the interface, as well as some more specific *do* or *don't* questions. For a list of the main questions asked during the test, refer to Appendix E. During the questioning, participants were motivated to give suggestions for the improvement of the interface by asking them open questions starting with *why* or *how*.

6.1.2- Results

Most participants (6 out of 7) found the interface quite hard to use initially, but found it easy to get used to it within minutes. The 7th found it easy to use from the start. Six participants found the interface fun to use, or at least more fun than the traditional interfaces they used. One even commented that this way, the music selection process is as fun as listening to the music is itself: "it's just like a game". All participants found having a visual overview of their music collection useful. Some usability issues were identified, most of which had to do with the presentation of the information: buttons were labeled with single letters instead of words, which made it hard to understand what they did. Also, it was not always clear what the colors in the visualization meant. Still, five participants liked the appearance of the artist map interface and would not change the design or the palette – only the legend.

All participants liked the idea of selecting music based on vague criteria. Especially the mood attribute was found useful. Tempo and year are not useful for everyone, as two of the participants indicated. Instead of year and tempo, they would like to have a magnet type that indicates the time since a song was last played or a magnet indicating song length. Six out of seven participants said that a strong link between the list and the artist map would be useful. Finally, four participants liked the idea of having a visual method to create playlists.

An interesting result from this qualitative test is that extensive functionality is not what most people are looking for in an interface for portable music players. They just want to be able to select music in a small amount of time. Although they like the customizable overview of a music collection in which a large amount of information is presented, most participants were not interested in extra features such as automated actions (i.e. changing the coloring attribute after zooming). Being able to change the skin of the interface was found more important. New features were only recommended if they help to find music that suits a given situation easier or faster. Even in the limited interface that was tested, some features were not used - even after they were explained. To validate this result, the usage of the coloring feature and the number of magnet types participants use will be measured in the final user test.

The identified usability problems and the suggestions done while discussing the interface were used to improve and extend the artist map interface, leading to the final prototype that was tested in the quantitative user test.

6.2- Quantitative user test

The setup of the quantitative user test better reflected the kind of usage our interface was designed for. Instead of working with a mouse and a large pc screen, participants got to use a touch-screen on which the interface had its actual size of 4 inches diagonally. In this test, music could be played from a moderately large collection of songs (2248). The design of this more elaborate test was taken from an evaluation of 'Satisfly' [41] and adapted for our purposes.

Apart from usability on a small screen, this test assessed user task performance, perceived ease-of-use and usefulness, usage of the available features, and user preference of an interface with the artist map in comparison with a list-only interface. Participants in the test were asked to select 8 or 10 songs using both methods (the list-based interface with and without the artist map) for a fixed music listening situation they personally described.

6.2.1- Hypotheses

Participants get all the time they need to select songs they want to hear; it is expected that the quality of the selected songs does not differ under the experimental conditions. The artist map should help them find music that matches their state of mind with less effort. Therefore, it is expected that:

1. Fewer actions are required to select songs when the artist map is used together with the list, than when only the list is used

When people browse large music collections – often with a large amount of unknown songs, it is expected that they tend to use simple strategies for selecting songs. I.e. they consider only a fraction of the available music, or restrain to habitual or random behavior. The artist map is designed to help people find music they did not know, so it is expected that:

2. Participants mostly select music from artists they like, but if they use the map they select more music from unknown, disliked or 'neutral' artists than they do with the list

As it is expected that the artist map helps in music exploration and saves effort without sacrificing the quality of the selected songs, participants will value the complete interface more on its usefulness than the list-only interface. This will result in an overall preference for the map. Still, learning to use the map takes some extra time initially. It is expected that:

3. Usefulness of the artist map is perceived higher than that of the list-only interface
4. Ease-of-use of the artist map is perceived lower than that of the list-only interface, because of the steeper learning curve for the artist map interface
5. Participants prefer using the map over only the list to find music that suits a situation

6.2.2- Participants

Twelve persons (9 male, 3 female) participated in the user test. Five of them were employees or students at Philips Research, while the other seven were people from outside Philips (mostly students). All participants were frequent listeners to popular music. The average age of the participants was 29 years (min: 20, max: 53). Table 6.1 shows how many artists people knew and liked, were neutral about, or did not like, out of a total of 111 artists.

	Know	Like	Neutral	Dislike
Average	75	28	34	13
Minimum	49	10	38	1
Maximum	94	57	7	30

Table 6.1: The average, minimum and maximum number of artists participants knew and liked, were neutral about, or did not like, in the test collection of 111 artists

Seven participants used portable music players in their personal life. Five of them used a Discman and five used an MP3 player. Half of the participants often played music from collections in which many of the songs were unknown to them. All of them used the same strategy to decide whether or not to play these unknown songs: they select a song at random, listen to a small part of it, and skip to the next song if they do not like it.

6.2.3- Test equipment and material

A music collection comprising 2248 popular music recordings from 111 different artists covering 7 different musical genres released in the period from 1963 to 2001 in MP3 format served as test material. The complete artist listing is shown in Appendix F.1. All music was stored on a central Linux server and transported via Ethernet for play back during the test. The test equipment consisted of a modern PC (Pentium 3) on which the system was running and a Fujitsu tablet on which only the front-end was running. The touch-screen of the tablet was used to control the system on the PC. The audio was directed to a Philips Streamium. Figure 6.1 shows two pictures of the test setup.

Figure 6.1: Pictures of the test setup, showing the Streamium, the PC and the Fujitsu tablet

Participants were seated in a comfortable chair, holding the Fujitsu tablet in their hands. They were asked to imagine that the interface they used on the tablet was actually the whole system (i.e. the portable music player). The experiment leader was located in the same room and was provided with a monitor linked up to the PC, allowing real-time observation and data collection of the task progress.

6.2.4- Procedure

The twelve participants were invited for a one-and-a-half hour experimental session in the WY building on the High Tech campus. A few days before admission to the test, participants were provided with a list of all artists that were used as music material in the experiment (see Appendix F.1). They were asked to indicate what artists they knew, and whether or not they liked to listen to them.

At the start of the session, participants were asked to complete a form to gather personal data (see Appendix F.2). Subsequently, they were handed over a sheet providing the general instructions of the experiment (see Appendix F.3). They were also asked to write down their personal listening environment for which they would like to select songs.

The participants received a sheet containing instructions on the use of the interactive system they were supposed to work with during the session (see Appendix F.4). After reading the instructions, they were presented a set of 10 small user tasks to perform. Participants were helped by the experiment leader, if needed. After performing the user tasks, participants were given the opportunity to practice with the system, until they felt comfortable with the system. They were allowed to ask questions on the use of the system to the experiment leader. After conclusion of the system practice, questions were no longer answered.

The experimental test consisted of two song selection tasks (selecting 8 or 10 songs with and without the artist map). For the first selection task, participants used only the list and for the second task they were allowed to use the artist map together with the list.

Participants were provided a sheet with nine rules they had to conform to while performing the tasks (see Appendix F.5). In short, the rules clarified that they were asked to select 8 or 10 different songs in two trials while imaging the same listening situation. The songs selected in the two trials should be different as well, and the number of songs selected in the trials should be equal. While performing the task, music could be listened to as many times as participants desired. No clues were given on how the task should proceed, or how music should be examined and evaluated. When the participants found a song they would like to add to their list, they were asked to write it down or tell the experiment leader. Time to perform the task was unlimited and speed of operation was not presented as a criterion of success. Quality of the selected songs was presented as the sole optimization criterion.

After each song selection task, participants completed a questionnaire assessing *perceived ease of use* and *perceived usefulness* of the interactive system (see Appendix F.6). At the end of the experimental session, participants ranked the two methods that they had used in the tasks according to their preference of use (see Appendix F.7). Also, they were presented with the two sets of songs that they had selected just a few moments earlier. They were asked to indicate which of these selections suited their listening situation better. To conclude the session, participants were asked to comment on desirable and undesirable properties of the system, and to tell what they would like to see added to the system (see Appendix F.8).

6.2.5- Measures

We measured task performance, selection of liked or other artists, perceived ease-of-use and usefulness, and user preference of either the list-only or the complete interface. In addition, we measured the usage of the available coloring option and magnet types.

Task performance
Task performance was measured by *number of actions*, by counting the number of basic touch-screen actions (i.e. clicks and scroll actions, where a single scroll action lasts up to 2 seconds) that were performed by the participant. *Time on task* and *rate of operation* were not measured because of the slow responsiveness of the artist map that limited the interaction speed, and because participants interacted with the experiment leader during the tasks. Note that questions about the interface were not answered anymore during the test.

Selection of liked artists
For both methods, we counted the number of times participants selected a song by an artist that they knew and liked as well as the number of times they selected a song by other artists.

Perceived ease of use and perceived usefulness
The Technology Acceptance Model (TAM) [42] defines two subjective terms related to usability and usefulness of an interactive system. In our experimental setting, the term *perceived ease of use* refers to the extent to which a user finds a user interface for a portable music player easy to learn and use. The term *perceived usefulness* refers to the extent to which a user finds such a user interface to be an aid for music selection. We modified the TAM questionnaire that was used in [41] to reflect the current domain (see Appendix F.6). Each term was assessed by posing five positive statements. Participants responded by stating to what extent they agreed with each statement in the questionnaire on a 7-point scale (1 = strongly agree,..., 4 = neutral,..., 7 = strongly disagree).

The statements assessing *perceived ease of use* were the following:

Q1. I find learning how to use the system easy.
Q2. I find it easy to get the system to do what I want it to do.
Q3. I find it easy to become skilful at using the system.
Q4. I find the system easy to use.
Q5. I find that by using the system I can easily select music that matches my state of mind.

The statements assessing *perceived usefulness* were the following:

Q6. I find that by using the system I am able to select music I want to hear rapidly.
Q7. I find that by using the system I enjoy selecting songs.
Q8. I find that the system helps me find unknown but enjoyable music.
Q9. I find the system useful for the situation I described.
Q10. I find the system useful for a portable device with a small touch-screen.

Order of preference
By having participants rank the interfaces and the selected songs using the questions listed below, *order of preference* of the interface systems was assessed (see Appendix F.7).

1. What system do you like *most*?
2. What system do you like *most* for selecting music in the situation you described?
3. For which method did you *like* the selected songs *better*?

Possible answers included the list-only interface and the list combined with the artist map, and the participants were asked to motivate their choice.

Feature usage
For the artist map method in this test, the usage of the coloring option and the number of magnet types each participant used was measured. The reason for this measurement is that the features were only used minimally in the qualitative test.

6.2.6- Results

Four participants described two listening situations instead of one, because they would use a portable music player in (at least) two different situations that asked for different kinds of music and they wanted to select songs for both. Instead of finding 8-10 songs once, they were asked to select 4-5 songs for each of their described listening situations. This change of the imagined listening situation did not affect the results.

Task performance

On average, the participants performed 328 actions when they used the list-only interface to select their songs, and 319 actions when the complete interface was used. See Chart 6.1.

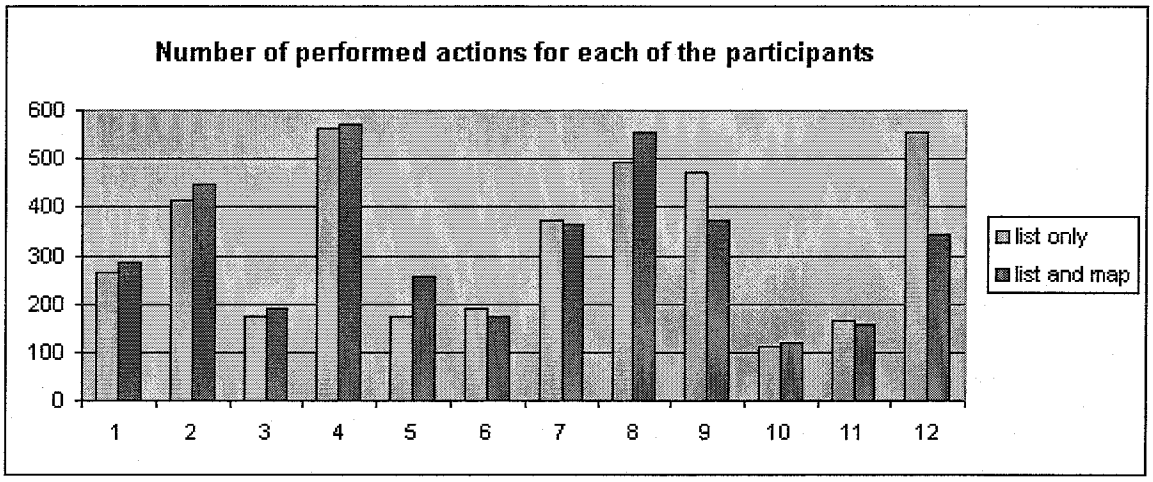


Chart 6.1: The amount of basic actions participants performed to select 8-10 songs. On average, the participants performed 328 actions with the list-only interface, and 319 actions with the list and map combined. However, only 5 of the 12 participants needed fewer actions with the complete interface than with the list-only interface

One of the reasons for the high amount of performed actions is that scrolling in the list takes a lot of time and thus a lot of actions (20 seconds of scrolling is seen as 10 actions). It is expected that if music can be played directly from the map, the difference in the amount of performed actions for the two methods will be more significant; the amount of actions in the map will be less. The reason is that for the list, scrolling will always be necessary, while in the map the time spent scrolling is replaced by a small number of actions (i.e. selecting artists or positions and clicking the play button) if music can be played directly. Chart 6.2 shows the amount of actions spent in the list when the complete interface was used, compared to the total number of actions.

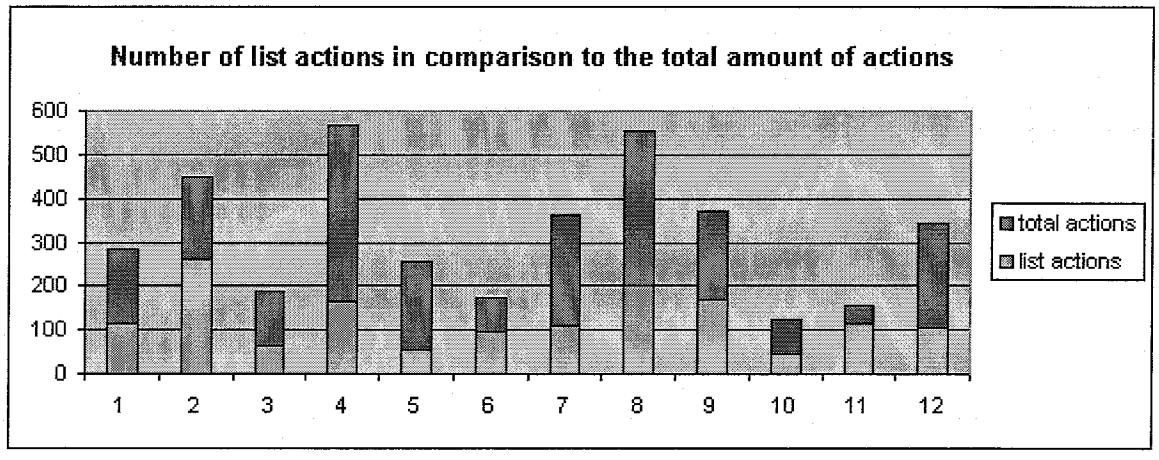


Chart 6.2: The amount of list actions in comparison to the total amount of actions participants performed when they used the complete interface. On average, 39% of the actions participants performed are still spent on scrolling and navigating the list

Selection of liked artists
For each of the selected songs, the participant's categorization of the corresponding artist was checked in both selection tasks. Participants indicated for each artist whether they liked the artist, were neutral about the artist, did not like the artist, or did not know the artist. Chart 6.3 presents the results. Participants mostly selected music from artists they liked, and selected more music from other artists with the artist map than without it.

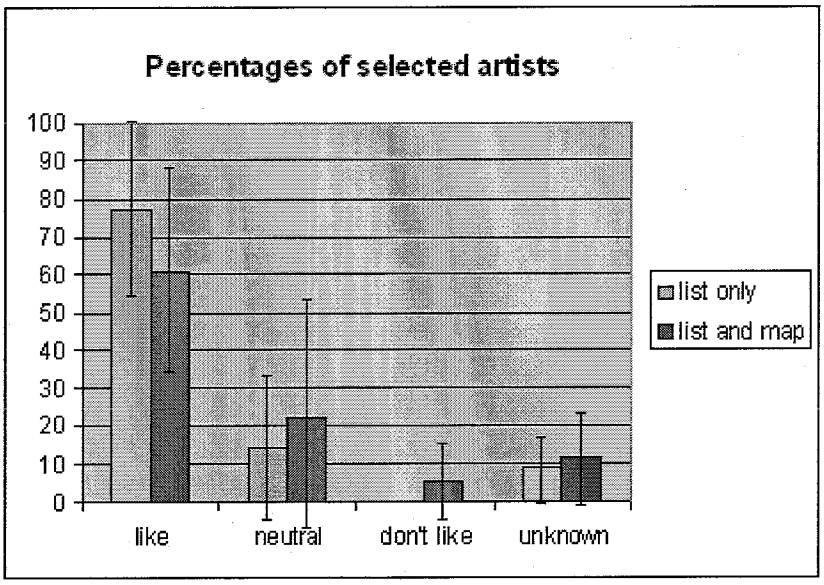


Chart 6.3: Average percentages and standard deviations for each of the four categories of the selected artists – for the list-only interface as well as the complete interface. Participants mostly selected artists they knew and liked. With the artist map they selected more neutral, disliked and unknown music than without it

Perceived ease of use and perceived usefulness
The TAM questionnaires that measured perceived ease of use and perceived usefulness resulted in 20 values for each participant: 5 values for ease of use and 5 for usefulness, for the list-only interface as well as for the complete interface. In Charts 6.4 and 6.5 the results of the TAM questionnaires are presented by showing the average result for each question and the corresponding standard deviations.

From Chart 6.4 it can be seen that participants perceived the list-based interface to be somewhat easier to use, although the complete interface made it easier to select music that suits their state of mind. The perceived ease of use of the artist map is lower than that of the list-only interface, because participants are used to list-based interfaces and the artist map therefore has a steeper learning curve. The most significant results from the perceived usefulness presented in Chart 6.5 are that the artist map helps to find unknown but enjoyable music much better than the list-only interface, and that the participants found selecting music with the map more fun than selecting music with the list-only interface.

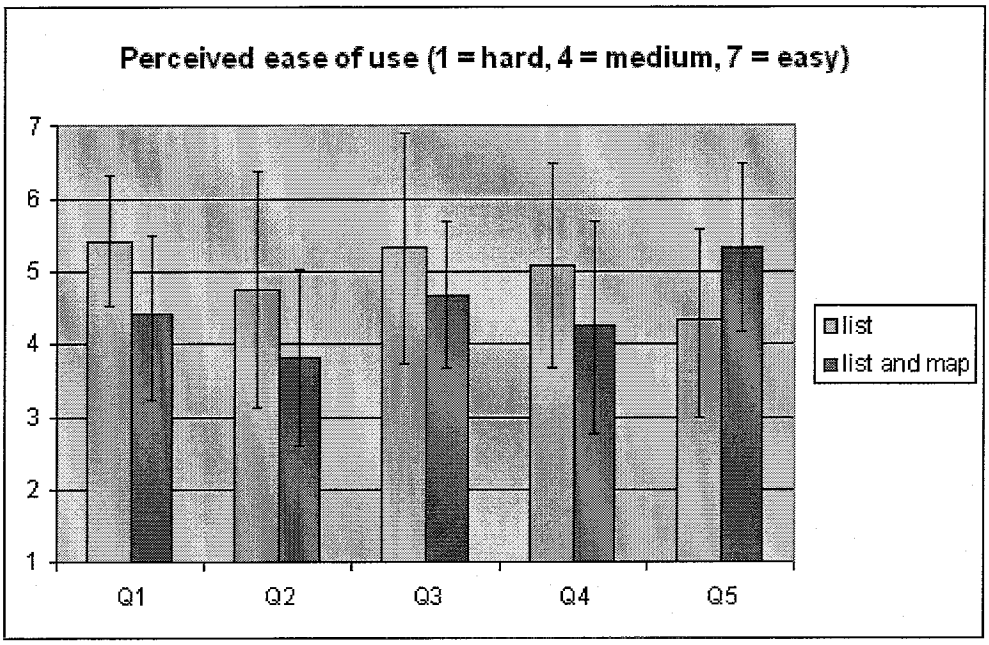


Chart 6.4: Perceived ease of use comparison of the list-only interface and the complete interface including the artist map. Overall, participants found the list easier to use than the complete interface, although the complete interface made it easier for them to select music that matched their state of mind (Q5)

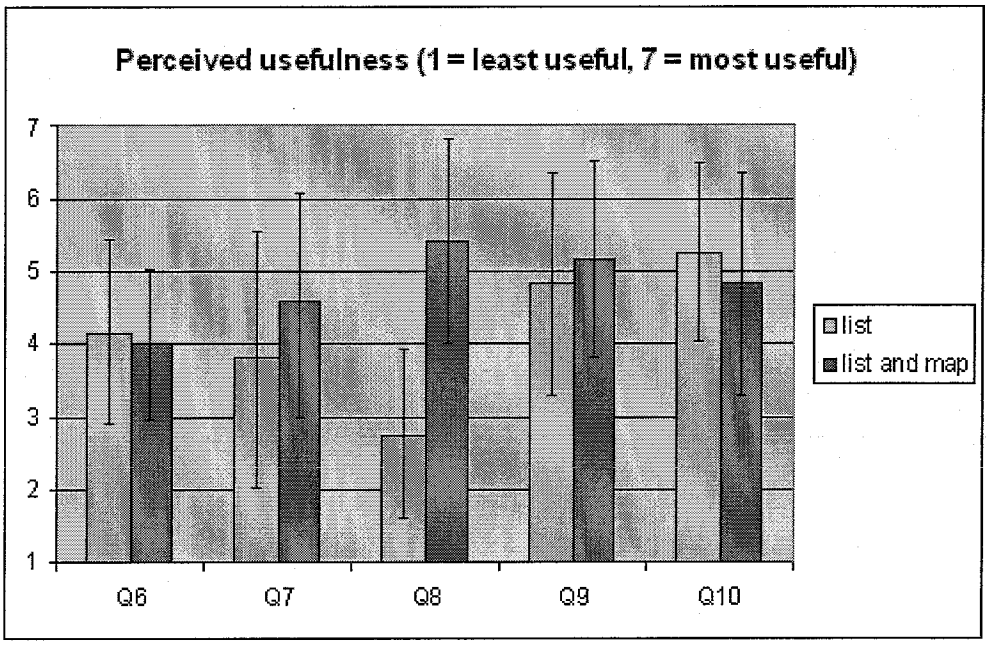


Chart 6.5: Perceived usefulness comparison of the list-only interface and the complete interface that includes the artist map. Participants found the list interface (slightly) more useful for rapid music selection (Q6) and better suited for a portable music player with a touch-screen (Q10). However, they find the artist map more useful for the situation they described (Q9). With the artist map participants enjoy selecting songs more (Q7), and the map helps them much better in finding unknown but enjoyable music (Q8)

Order of preference
Participants were asked three questions about their preference regarding the two different methods: one about overall usage, one about the particular situation, and one about the resulting sets of songs they selected. The results are presented in Charts 6.6, 6.7 and 6.8.

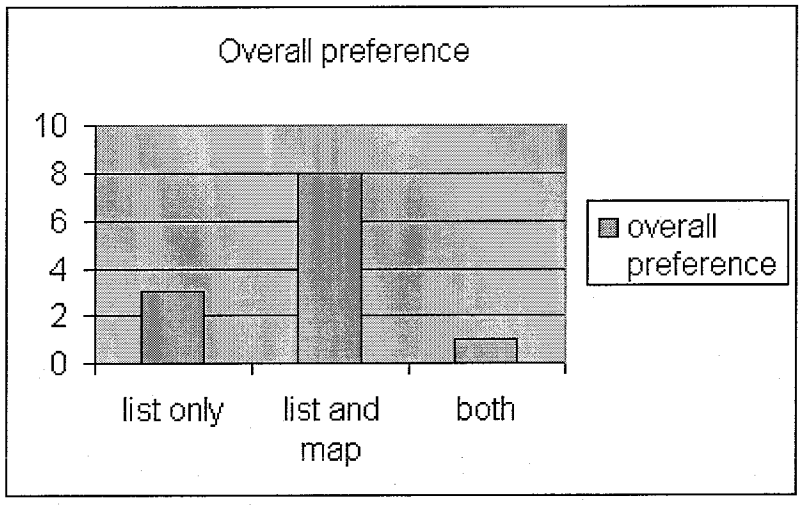


Chart 6.6: Number of participants with an overall preference for the list-only interface, the complete interface with the artist map, or both. Most participants liked the list and map best

For the overall usage, one of the participants could not choose and liked both methods equally well, 8 participants liked the list combined with the artist map best, and 3 participants liked the list-only method best. The participants gave the following reasons for liking the combined method better:

- The map helps you find unknown but possibly enjoyable music (3 times)
- In the map you can find music that suits a certain mood (3 times)
- The overview makes it easier to find music, especially in a large collection (3 times)

The reasons they gave for liking the list-only interface better were:

- The list is familiar and makes it easy to play specific and well-known songs (3 times)
- The map is too slow and does not let me play music directly (1 time)

For the situation they described, 6 participants liked the list combined with the artist map best, 2 liked the artist map and another method equally well (using their own Discman, random and skipping), 2 liked the list-only method best, and 2 liked their current music player best. The reasons for liking either the combined interface or the list-only interface better are similar to the reasons stated above. The motivation that participants most often gave for liking another method (i.e. their own Discman or I-Pod) better than the designed interfaces, was:

- Selecting songs in the described situation should be very fast and easy (2 times)

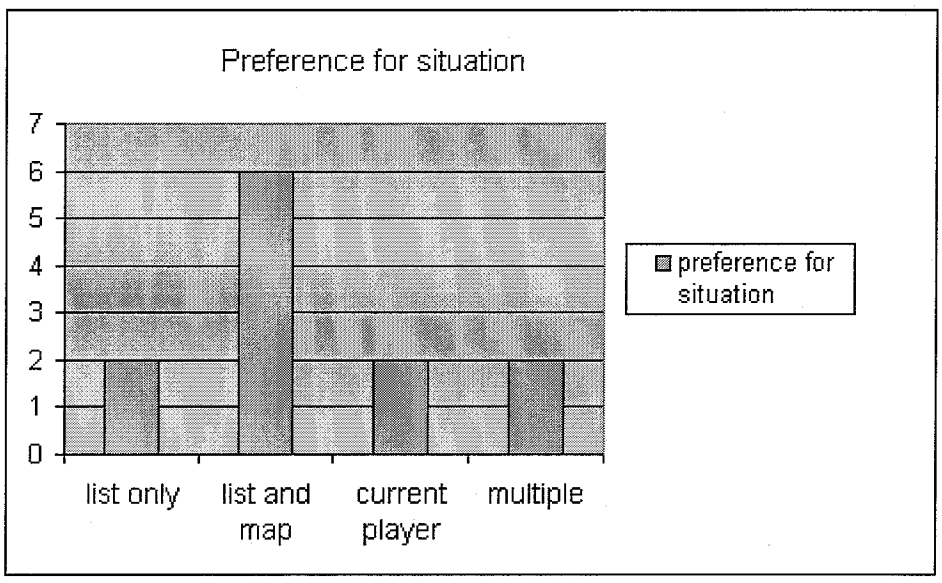


Chart 6.7: Number of participants with a preference for the list-only interface, the complete interface with the artist map, the interface of their current portable music player, or more than one of the above, for the situation they described. Most participants liked the list and map best

The two sets of songs that participants selected were mostly liked equally well (7 participants said so). Three participants liked the songs they selected with the combined method better, and two participants liked their songs selected with the list-only method better – see Chart 6.8.

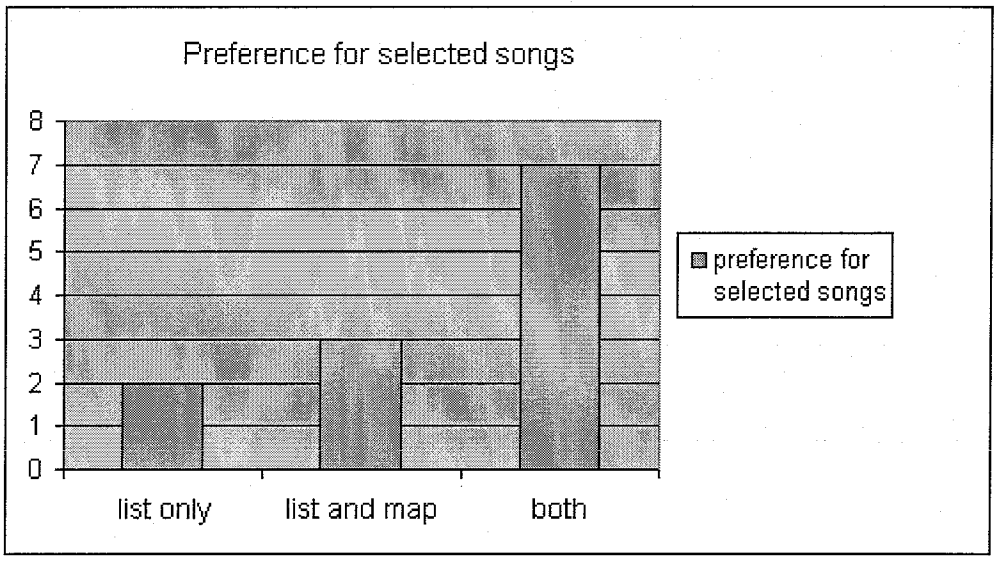


Chart 6.8: Number of participants with a preference for the songs they selected with the list-only interface or the complete interface with the artist map, or both. Most participants liked both sets equally well

Feature usage
Although the existence of the coloring feature was explicitly pointed out in the experimental phase of the test, only five participants actually used it when they selected their songs. Out of these five, two participants used it only once. One third of the participants did not use or even try all of the available magnet types. Four participants did not functionally move the magnets on the screen to create groups; they only moved magnets slightly a few times (by accident).

Liked and disliked system properties
Participants were asked to comment on the desirable and undesirable properties of the complete user interface at the conclusion of the experiment. Eleven participants gave comments on multiple system aspects. Especially when it came to commenting on disliked system properties, usability issues of the interactive part of the system were addressed (most notably the interaction speed). Although this experiment was not a usability test, the comments provided valuable insight in the usability aspects of the total concept. Below we summarize the most important or most frequent comments.

Liked properties
Six out of the twelve participants explicitly commented that they liked the ability to find music based on 'kinds' of music. They were especially positive about the mood attribute.

Three participants said that they like the overview the map and its magnets provide. One of the benefits is that it helps them find music they did not know.

Two participants commented that having a small interface (having everything on screen at once) for large music collections is beneficial, and gets more important as music collections grow.

One participant noted that the coloring of artists based on an attribute helps in remembering what kind of music you like – remembering colors is easier than remembering names.

Disliked properties
Seven participants commented that the interface is slow, and all twelve participants made at least one remark about it during the test.

Four participants commented that the interaction with the list is not as they expect, i.e. selecting a single song and clicking play should always result in playing that song. Also, it is not always clear to them which items are selected in the list.

Two participants commented about the size of labels and artists in the map, which sometimes made the map less clear than it should be.

Suggestions
In the final question of the user test, participants were asked to suggest additions to the artist map interface. Below we list the most important or most frequent suggestions.

- Being able to create a playlist (mentioned 5 times)
- Fast forward functionality during playback (mentioned 2 times)
- Create a stronger link between the map and the list, i.e. it would be interesting to seed the map with an artist selected in the list (mentioned 3 times)
- Visualize the entire genre-artist-album-song hierarchy in the buttons above the list interface, so it is easier to see where you are (mentioned once)

6.2.7- Discussion

This experiment evaluated user task performance, selection of liked or other artists, perceived ease-of-use and usefulness, and preference of use of the complete artist map interface in comparison with a list-only interface. Participants were not informed about the purpose and design of the test. Also, they were not instructed or limited on how to perform the song selection task using both methods.

It was expected that fewer actions are required to select songs when using the artist map interface than when using the list-only interface. The test found out that participants needed 9 actions less to do a similar task using the artist map. The difference in the amount of required actions is expected to increase significantly if music can be played directly from the map, since the amount of scrolling and navigating actions that participants performed was very large: 39% of their actions were spent on scrolling and moving back and forth in the list. Note that selection of items and using the music playback buttons were not included in this set of list-actions. Based on these results, the hypothesis could not be rejected. In short, the artist map enabled the participants to create their preferred playlist in fewer actions – and this result is expected to be easily improved in the future by enabling music playback directly from the map.

It was expected that most of the selected songs are from liked artists, but if the artist map is used, more songs from unknown, disliked or 'neutral' artists are selected than there are when only the list is used. The test found out that participants mostly selected songs from artists they liked in both cases. When the artist map was used, however, participants selected more songs from neutral and unknown artists than when they used the list-only interface – see Chart 6.3. They even selected songs from artists they thought they did not like when they used the map. Based on these results, the hypothesis could not be rejected. Note that the participants were asked to select songs that fit their described listening situation, and that they had to like the songs they selected. Also note that most participants, 10 out of 12, did not find the songs they selected with the list-only interface better than the songs they selected with the artist map. In short, this means that participants discover new music that they like to hear with the artist map.

It was expected that the usefulness of the artist map interface was perceived higher than the list-based interface, while the ease-of-use of the artist map was perceived lower. The TAM questionnaire indicated that the artist map interface was perceived most useful, especially for finding unknown but enjoyable music, and that the list-based interface was found most easy-to-use (the artist map was only perceived easier to use to find music that suits the state of mind of the listener). Based on these results, the hypothesis could not be rejected. In short, the artist map was found more useful but less easy to use than the list-only interface. One of the reasons participants find the list-only interface easier to use is that they are used to this kind of interface.

It was expected that the use of the artist map interface is more preferred than the use of the list-only interface. The test found out that the artist map interface was definitely more preferred for overall use as well as in the situation the participants described. Based on these results, the hypothesis could not be rejected. In short, participants preferred to use artist map instead of the list-based interface.

The most significant results of the user test show that the artist map helps people find unknown but enjoyable music. People even like the newly discovered music at least as well as the songs they normally play using list-only interfaces. The results also indicate that people like a visual music selection method better than a list-only interface, even though it may take some time to get used to it. Therefore, we feel that research in this field should be continued and we expect that a visual interface will be an important selling point for portable music players of the future.

7- Conclusions and future work

In this final chapter, our realized work is compared to the requirements to find out if we succeeded in meeting our goals. Based on user tests, we discuss why we feel that the world is ready for a new kind of visual interface for portable music players, based on the artist map interface. Finally, some directions for future research will be given.

7.1- Conclusions

The artist map provides a music overview that users find quite easy to understand. In this overview, similar artists are positioned close together. To provide contextual information in the overview, the attributes *mood*, *genre*, *year* and *tempo* are used as magnets that influence the position of artists in the map. These four attributes closely follow the personal descriptions people tend to give when they describe music they would like to hear.

Fuzzy as well as specific searches are supported by our interface: fuzzy search criteria can be used in the artist map to find a certain kind of music, while a specific song, artist or album can be found by using the list. In the artist map, the user determines what kind of music goes where, since the user has control over the magnets.

We conclude that the artist map covers most of our requirements. Unfortunately, visual playlist creation has not been realized, although we did make a conceptual start for such a feature.

From the user tests, we have learned that the list will not be replaced by a purely visual high-level music selection method in the near future, because people always want to be able to select specific songs. However, the artist map provides extra functionality that people find useful, especially for large music collections. An important conclusion resulting from the user tests is that a visual interface helps people to find other music than they normally listen to (with fewer actions), while they find this music at least as good. Also, using the artist map enhances the fun of selecting music: people enjoy the process itself instead of finding it a necessary evil.

Based on these findings and the perceptions of participants in our tests on the usefulness of the artist map, we expect that people are willing to learn a new kind of interface for their portable music players: a visual method based on the artist map. The reward of using such an interface seems to be well worth the steeper learning curve and initial effort. Furthermore, it seems worthwhile to investigate this kind of interface further given the positive test results.

7.2- Future work

All participants in the user tests commented on the responsiveness of the artist map interface: it is too slow. An obvious recommendation for the future is to improve the speed of the interface and to test it again: people may well be even more positive about the artist map if the speed of the interface is acceptable. The responsiveness of the interface can be improved by optimizing the graph layout algorithm: i.e. the magnets may be used to improve the speed of convergence. Obviously, speed can also be gained by running the system on a single machine (as opposed to the current distribution over a pc and a touch-screen tablet) without using scripting or interpreted languages. Other recommendations for future extensions are the following:

- Playlist creation – Visual playlist creation in the map as well as playlist creation functionality in the list should be added to the interface. Combining both features has the added benefit that a playlist can be seeded by vague criteria encoded in the playlist path drawn on the map, while it can be finished with more specific requirements in the list.

- Playing music directly from the map – If music could be played directly from the map, less interaction would be required to play music that people want to hear. Switching between the map and the list would no longer be needed to select and play enjoyable music, although the list would still be useful to select individual songs.

- Learning capabilities – Currently, the artist map interface uses the mood, genre, year and tempo of songs as well as the more standard catalogue metadata. It would be interesting to see the added benefit of extra metadata that is learned from the behavior of the user. For instance, the interface could learn which songs the user often plays, or which songs are often played in the same sequence. More basic 'learned' metadata such as the date last played or the number of times played could also be used in the interface, i.e. by creating a new magnet type for such metadata.

- Stronger connection of list and map – It would be beneficial to integrate the list and the map more strongly. Selecting songs or artists in the list and seeing where they are on the map, or which similar items there are on the map is something that many people would like to do, as we learned during the user tests.

- Improved use of the touch-screen – The artist map interface contains many buttons, while the interface is to be used on a touch-screen. Instead of using these standard buttons on the interface, it would be interesting to experiment with alternative interaction methods. For example, gestures could be used. Instead of clicking on the map and seeing a fixed-size zoom-region, the user could also *draw* the zoom-region. Dragging upwards on the screen after drawing the region could mean zoom out, while dragging downwards could mean zoom in. A disadvantage of such alternative interaction methods is that they will probably make the learning curve steeper.

References

[1] B. de Ruyter, H. Sinke, S. Pauws, R. Salpietro, Easy Access to Networked ICE: User issues in accessing large collections of audio content, Philips Internal Report PR-TN-2000/00145, 2000

[2] Windows Media Player. *www. microsoft. com/windowsmedia*, 2004

[3] Music Match. *www. musicmatch. com*, 2004

[4] B. Vaessens, Expression of music preferences: how do people describe their pop music, MSc Thesis, University of Tilburg, Philips Internal Report PR-TN-2003/00016, 2003

[5] F. Vignoli, A. Bondaryeva, A user study to identify novel concepts to browse and navigate through digital music collections, Philips Internal User Interfaces conference, Philips Internal Report PR-TN-2003/00717, 2003

[6] WordIQ, definition of mp3, *http://www.wordiq.com/definition/MP3*

[7] Encyclopedia website, music file formats,
http://encyclopedia.thefreedictionary.com/Music+file+format

[8] A. Berenzweig, B. Logan, D. Ellis, and B. Whitman, A Large-Scale Evaluation of Acoustic and Subjective Music Similarity Measures. *Proceedings of 4th International Conference on Music Information Retrieval, ISMIR 2003*, pp. 99-105, 2004

[9] R. Spence, Information Visualization, Pearson Education, 2000

[10] C. Ware, Information Visualization: Perception for Design, Morgan Kaufmann, 2000

[11] Sequoia View website, *http://www.win.tue.nl/sequoiaview/*

[12] J. Wise, J. Thomas, K. Pennock, D. Lantrip, M. Pottier, A. Schur, V. Crow, Visualizing the Non-Visual: Spatial Analysis and Interaction with Information from Text Documents, *Proceedings of the IEEE Symposium on Information Visualization*, pp. 51-58, 1995

[13] F. van Ham, Private communications about hierarchical graphs research in his PhD project, 2003

[14] VisDB homepage, *http://www.dbs.informatik.uni-muenchen.de/dbs/projekt/visdb/visdb.html*

[15] G. Di Battista, P. Eades, R. Tamassia, and I. Tollis, Graph Drawing: Algorithms for the Visualization of Graphs, Prentice Hall, 1998

[16] Archos AV series, *http://www.archos.com/*

[17] Apex MP series, *http://apexdigitalinc.com/*

[18] IRiver PMP series, *http://www.iriver.com/*

[19] R. van Liere, W. de Leeuw, GraphSplatting: Visualizing Graphs as Continuous Fields, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 9, No. 2, pp. 206-212, 2003

[20] F. van Ham, Interactive Visualization of Small World Graphs. Accepted for publication at the *IEEE Symposium on Information Visualization*, 2004

[21] MoodLogic. *http://www.moodlogic.com*, 2004

[22] A. Cruts and H.C.M. Hoonhout, The color of music. Looking for a consistent association between music features and colors, Philips Internal Report PR-TN-2003/00553, 2003

[23] M. McKinney and J. Breebart, Features for Audio Music Classification. *Proceedings of 4th International Conference on Music Information Retrieval, ISMIR 2003*, pp. 151-158, 2003

[24] D. Liu, L. Lu, and H-J. Zhang, Automatic mood detection from acoustic music data. *Proceedings of 4th International Conference on Music Information Retrieval, ISMIR 2003*, pp. 81-87, 2003

[25] G. Tzanetakis, and P. Cook, Musical Genre Classification of Audio Signals. *IEEE Transactions on Speech and Audio Processing*, Vol. 10, No. 5, pp. 293-302, 2002

[26] B. Logan, Mel Frequency Cepstral Coefficients for Music Modeling. *Proc. of International Symposium on Music Information Retrieval*, pp. 23-25, 2000

[27] K. Fukunaga, Introduction to Statistical Pattern Recognition, Computer Science and Scientific Computing Series, Academic Press, 2nd edition, 1990

[28] T. Kohonen, Self-Organizing Maps, Springer-Verlag, 3rd edition, 2000

[29] M. Qasem, X. Du, S. Ahalt, On Efficient Codebook Search for Vector Quantization: Exploiting inherent Codebook Structure, *Proceedings of SPIE*, Vol. 3716, pp. 149-154, 1999

[30] L. Talavera and J. Bejar, Generality-Based Conceptual Clustering with Probabilistic Concepts, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 23, No. 2, pp. 196-206, 2001

[31] P. Eades, A heuristic for graph drawing. *Proceedings of Congressus Numerantium*, Vol. 42, pp. 149-160, 1984

[32] S. Kirkpatrick, C. Gelatt, and M. Vecchi, Optimization by Simulated Annealing, Science, Vol. 220, No. 4598, pp. 671 – 680, 1983

[33] T. Fruchterman, E. Reingold, Graph Drawing by Force-directed Placement. *Software - practice and experience*, Vol. 21, pp. 1129-1164, 1991

[34] M. Kaufmann, D. Wagner, editors, Drawing Graphs: Methods and Models, *Lecture Notes in Computer Science*, Vol. 2025, Springer, 2001

[35] A. Noack, An Energy Model for Visual Graph Clustering. *Proc of 11th International Symposium on Graph Drawing,* pp. 425-436, 2003

[36] A. Noack, Energy Models for Drawing Clustered Small-World Graphs, Computer Science Report, BrandenBurg Technical University at Cottbus, 2003

[37] R. van Gulik, F. Vignoli, H. van de Wetering, Mapping music in the palm of your hand, explore and discover your collection, Accepted for publication at *the 5th International Conference on Music Information Retrieval, ISMIR 2004*, 2004

[38] Qt website, *http://www.trolltech.com*, 2004

[39] Macromedia Flash website, *http://www.macromedia.com/software/flash/*, 2004

[40] F. Vignoli, Mma: architectural framework and algorithms to support multiple input modalities, Philips Internal Report PR-TN-2003/00663, 2003

[41] S. Pauws, Evaluating playlist creation with SatisFly assistance, Philips Internal Report PR-TN-2003/00714, 2003

[42] F.D. Davis, Perceived usefulness, perceived ease-of-use, and user acceptance of information technology. *Management Information Science Quarterly,* Vol. 18, 189-211, 1989

A- Project Description

Graduation project: "Graphical user interface for Portable HDD players"
Context: Philips Natlab, Software Architectures group,
Personal Infotainment Companion (PiC) project
Starting date: September 8th 2003

Portable hard disk players like the Apple iPod and the Philips HDD100 are a type of music player. With a storage capacity of literally 1000's of songs they will soon be capable of storing your entire music collection. It becomes problematic to navigate this large collection on the pocket-sized display. Philips is interested in experimenting with user interfaces for music collections that fully exploit the available meta-data (genre, artist, album, year) and go beyond the traditional directory structure.

The graduation project consists of two tracks:

1) A research track, where a new visualization technique for MP3 collections on a small screen will be developed. Preferably, the visualization technique enables easy & fun browsing of a large music collection and provides the user with a new way to create play lists – making it easy to find interesting music without searching for specific songs.

2) An implementation track, where the available C++ PiC prototype code-base is extended with a Qt interface, and the experimental visualization technique of 1) is built in C++.

The graduation project builds on the PiC prototype code-base that is available at the Natlab. The code provides a C++ API that allows access to both the local MP3 collection, and remote collections that can be reached over a multitude of the (wireless, wired) networks.

Project phasing:

1) **Familiarizing**. Both tracks proceed in parallel. The student becomes familiar with visualization techniques, the C++ API of the PiC prototype & learns about Qt and OpenGL. [1 month]

2) **Analysis**. After the initial learning phase, the student will design a number of new visualization techniques for a large music collection & make an informed decision which technique to implement. He will also design and implement a first Qt interface on which the visualization technique can be utilized [1.5 months]

3) **Design**. The student improves the level of detail of the chosen design, and specifies how the visualization can be used as an interface that provides a PiC user with fun and easy to use browsing capabilities. [1 month]

4) **Implementation**. The student implements the designed visualization technique (along with its user interface capabilities) on top of the Qt interface & goes to some iterations to improve the visualization technique. The current scheduling provides this phase with extra time, because it might be needed to reconsider some visualization aspects [2.5 / 3 months]

After each phase, up to 2 weeks is reserved for writing down the achieved results. This leaves an estimated 1.5 months at the end of the project to finalize the thesis.
The emphasis of the project is on visualizing a large collection of music and building a user interface using this visualization. Extending this interface with extra functionalities such as a recommendation system (a secondary objective) will only be considered if time allows it.

B- Correlation Experiments

This appendix describes the results of the correlation experiments that were done to find suitable colors for the genres used in the artist map. Since suitable colors were already found for tempo and mood, we looked for correlations between genre and tempo and between genre and mood in our test music collection of 2248 songs. Tables B.1 and B.2 show the results of these correlation experiments.

Genre \ Tempo (in BPM)	Very slow (<68)	Slow (68-80)	Medium (80-120)	Fast (120-155)	Very fast (>155)
Rock	4.4	19.8	41.6	21.6	12.7
Popular	2.7	19.7	47.7	24.6	5.3
Americana	6.1	21.2	44.3	19.8	8.5
Alternative	7.8	19.0	47.5	16.2	9.5
Soul/R&B/Rap	2.1	15.9	66.9	12.5	2.4
Dance/Lounge	11.2	17.7	44.4	16.2	10.5
Nederlandstalig	7.1	21.5	40.7	18.9	11.8

Table B.1: Relative tempo distributions for songs in our test collection, given the genre. 4.4% of all Rock music in the collection has a very slow tempo (<68 beats per minute). The tempo ranges have been chosen in accordance with studies on the perceived tempo of music.

Genre \ Mood	Upbeat	Happy	Romantic	Mellow	Sad	Sentimental	Brooding	Aggressive	Unknown
Rock	29.1	11.3	13.8	4.2	9.6	6.4	9.6	9.3	6.7
Popular	34.1	14.4	17.0	7.2	11.7	5.3	5.7	3.8	0.8
Americana	13.7	9.4	14.2	9.9	22.2	14.6	11.3	1.9	2.8
Alternative	13.3	8.8	6.4	6.2	14.0	14.7	16.4	14.5	5.7
Soul/R&B/Rap	34.3	18.7	13.5	6.1	6.7	4.9	7.6	4.6	3.7
Dance/Lounge	18.1	13.0	19.1	5.8	9.0	13.4	11.2	6.5	4.0
Nederlandstalig	5.1	6.1	2.7	1.3	5.1	7.4	0.7	1.3	70.4

Table B.2: Relative mood distributions for songs in our test collection, given the genre. 29.1% of all Rock music in the collection is associated with an upbeat mood

From the results we see that most music is of medium tempo (80 < BPM <= 120), no matter which genre we look at. The reason is that the music in our test-collection is mainstream. Coloring everything in a neutral color such as green or brown, however, does not seem like a good idea. To keep genre colors easily distinguishable, we keep medium tempo influences out of the picture for the most part. The only exception is Soul / R&B / Rap, for which over 65% (!) is of medium tempo in our music collection: this genre will be associated with a neutral color.

- Nederlandstalig is a special case. For most of the songs, the associated mood is unknown, so linking a color to it on that basis is very hard. Tempo also varies a lot, and slow and fast are nicely balanced. However, orange is the color associated with Holland. As such, we also associate it with our music genre Nederlandstalig.
- Rock is mainly upbeat with a more often fast than slow tempo. This associates Rock with yellow, orange and red for the tempo, and bright red for the mood. Since orange is already taken and red is ok for both the tempo and the mood, we choose bright red for Rock music.

- Popular music in our collection often has a high tempo and has an upbeat, happy or romantic mood. This gives the genre both red and bright blue influences, leading to a purple or pinkish color.
- Americana is the only genre for which sad is the best matching mood in our music collection. This makes Americana a good candidate to get the color dark blue. The tempo that varies a lot in this genre does not change this color preference.
- The genre of Alternative music is the only one with high numbers in the brooding and aggressive departments. The fact that slow and fast Alternative songs occur equally often in our collection makes green a suitable color for Alternative music, but the moods push the color choice towards dark yellow, orange or red. Since red and orange are already taken in a brighter form, we choose dark yellow (which has some green influences) as the color of Alternative music.
- Soul, R&B and Rap has the highest percentage of medium tempo songs of all, which points to a neutral color choice. The mood, however, is also clearly distinguishable as being mostly upbeat or happy (red). We pick brown as the neutral color of choice, because brown contains a lot of red.
- Dance / Lounge music has many different moods and tempos associated to it. Upbeat and romantic, but also sentimental and brooding apply to this genre. The tempo does not push us toward a clear color either (slow and fast are equally well represented). Since green has not been used yet and Dance / Lounge music seems to be a perfect candidate, this genre will get a bright green color.

C- Similarity is not transitive

In this appendix we prove that transitivity does not hold in general for our similarity relation. Transitivity in the case of artist similarity means that if A is similar to B and B is similar to C that A is also similar to C, but unfortunately this is not necessarily the case, as we prove below.

For three artists A, B and C, artists A and B as well as B and C can be similar, while A and C are not. We prove this by constructing histograms for these artists, forming a counterexample for transitivity. Let us create a P×Q histogram, P > 1 and Q > 1, for each of these artists such that Sim(A,C) < **t** while Sim(A,B) ≥ **t** and Sim(B,C) ≥ **t**. A histogram H_I for artist I ($I \in \{A, B, C\}$) has elements $H_I(p, q)$, where $0 \le p < P$ and $0 \le q < Q$.

As an example, consider the case that a 2x2 histogram is used for similarity and the threshold **t** is equal to 0.6. The following matrices form a counter-example for transitivity:

$$H_A = \begin{pmatrix} 0.5 & 0.5 \\ 0 & 0 \end{pmatrix}, H_B = \begin{pmatrix} 0.5 & 0.25 \\ 0.25 & 0 \end{pmatrix}, H_C = \begin{pmatrix} 0.5 & 0 \\ 0.5 & 0 \end{pmatrix}$$

For these histograms, Sim(A,B) = 0.75 ≥ 0.6, Sim(B,C) = 0.75 ≥ 0.6, and Sim(A,C) = 0.5 < 0.6. Therefore, A and B are similar, B and C are similar, but A and C are not similar.

Although this already proves that artist similarity is not transitive in general, counterexamples can be constructed regardless of the exact threshold (as long as $0 < \mathbf{t} < 1$, because for the extremes **t** = 0 and **t** = 1 transitivity *does* hold) and histogram size, as we will show now. Since the histograms are at least 2x2 in size and these four indices are enough make the counterexample as seen in the example above, we disregard the rest of the histogram. Thus, for all artists I, for p > 2 and q > 2, we have $H_I(p, q) = 0$. To fill the rest of the histograms (the upper left 2x2 part of H_I is called $H_{I\text{-}0,0\text{-}1,1}$) we introduce a small delta ($\delta < \mathbf{t}$):

$$\delta = \mathbf{t} - \mathbf{t}^2$$

Delta is used to keep Sim(A, C) just below the threshold (i.e. $Sim(A,C) = \mathbf{t} - \delta = \mathbf{t}^2$), while Sim(A, B) and Sim(B, C) are high enough. The rest of histogram H_a, H_b, and H_c is filled as follows:

$$H_{A\text{-}0,0\text{-}1,1} = \begin{pmatrix} \mathbf{t}-\delta & 1-(\mathbf{t}-\delta) \\ 0 & 0 \end{pmatrix}, H_{B\text{-}0,0\text{-}1,1} = \begin{pmatrix} \mathbf{t}-\delta & (1-(\mathbf{t}-\delta))/2 \\ (1-(\mathbf{t}-\delta))/2 & 0 \end{pmatrix}, H_{C\text{-}0,0\text{-}1,1} = \begin{pmatrix} \mathbf{t}-\delta & 0 \\ 1-(\mathbf{t}-\delta) & 0 \end{pmatrix}$$

Or, substituting $\mathbf{t} - \mathbf{t}^2$ for δ:

$$H_{A\text{-}0,0\text{-}1,1} = \begin{pmatrix} \mathbf{t}^2 & 1-\mathbf{t}^2 \\ 0 & 0 \end{pmatrix}, H_{B\text{-}0,0\text{-}1,1} = \begin{pmatrix} \mathbf{t}^2 & (1-\mathbf{t}^2)/2 \\ (1-\mathbf{t}^2)/2 & 0 \end{pmatrix}, H_{C\text{-}0,0\text{-}1,1} = \begin{pmatrix} \mathbf{t}^2 & 0 \\ 1-\mathbf{t}^2 & 0 \end{pmatrix}$$

All three histograms are valid, since the sum of their elements equal one:

$$\sum H_A(p,q) = \mathbf{t}^2 + 1 - \mathbf{t}^2 = 1$$

$$\sum H_B(p,q) = \mathbf{t}^2 + (1 - \mathbf{t}^2)/2 + (1 - \mathbf{t}^2)/2 = 1$$

$$\sum H_C(p,q) = \mathbf{t}^2 + 1 - \mathbf{t}^2 = 1$$

But with these histograms:

$Sim(A,B) = \mathbf{t}^2 + (1 - \mathbf{t}^2)/ 2$
$Sim(B,C) = \mathbf{t}^2 + (1 - \mathbf{t}^2)/ 2$
$Sim(A,C) = \mathbf{t}^2$

Clearly, A and C are not similar, since $\mathbf{t}^2 < \mathbf{t}$, for $0 < \mathbf{t} < 1$.
But A and B as well as B and C are similar:

$\mathbf{t}^2 + (1 - \mathbf{t}^2)/ 2$

$= \frac{1}{2}\mathbf{t}^2 + \frac{1}{2}$

$> \mathbf{t}$

(Since $\frac{1}{2}x^2 - x + \frac{1}{2}$ describes a function that lies above the x-axis on the interval $0 < x < 1$)

This proves that the similarity relation is not transitive. Along the same lines it can be proven that A and C can be similar while A and B are not, and B and C are not either, i.e. $Sim(A,B) < \mathbf{t}$ and $Sim(B,C) < \mathbf{t}$, while $Sim(A,C) \geq \mathbf{t}$.

D- Settings for layout

The final settings used in the graph layout algorithms are as follows:

- Temperature = 1.5
- Temperature step = 0.004
- Switch temperature = 1.0
- Iterations for each step = 25
- Optimum distance (spring length) = 1.5
- Spring stiffness = 0.004
- Repulsion force = 0.07
- Similarity threshold (**t**) = 0.67
- Magnet attraction multiplier = 3

Switch temperature may need some explanation: The temperature is lowered after every step (a main iteration) of the graph layout algorithm. After the temperature has descended to lower than the switch temperature, distance computations are changed. Above the switch temperature, distance is measured as the distance between the midpoints of two vertices, while below the switch temperature distance is measured as the distance between the outer bounds of the vertices. This is to ensure that vertices do not hinder each other when they are moving toward their 'desired place' in the layout, but still make sure that vertices do not overlap too much when they have reached their approximate destination.

E- Qualitative user test

In the qualitative user test, the following questions were asked:

1. Do you like the interface visually?
2. Do you find the interface easy to use?
3. Do you think you can find music more efficiently with this interface (compared to lists)?
4. Do you think you can find music more effectively with this interface (compared to lists)?
5. Do you feel in control when you are using the interface?
6. Would you remove the color button and make the legend clickable instead?
7. Would you randomly show the names of artists on the screen?
8. Would you want the coloring to change automatically over time?
9. Would you like magnets to be automatically placed after zooming?
10. Would you want the coloring function to skip an attribute if it is homogenous?
11. Would you like (and use) graphical playlist creation functionality?
12. After zooming, would you expect the list to show only the artists that were in view in the map, or would you expect all artists to be in the list, though graphically distinguishable?

Note that all of these questions were followed up by open-ended questions such as *why* and *how*, and that the questions were asked during an informal test with much conversation.

F- Quantitative user test

This appendix contains all the forms that were used for the quantitative user test.

F.1- Artist overview for experiment

The experiment in which you will participate is about selecting songs from a large collection of pop music. The complete collection contains 169 albums from 111 different artists covering 7 different musical genres. Can you please indicate which artists you know and whether you like to listen to them by reading through the list, and bring the completed list with you at the day of the experiment?

ArtistNr	ArtistName	Know, like	Know, neutral	Know, don't like	Don't know
1	16 Horsepower				
2	2pac				
3	Abba				
4	Acda En De Munnik				
5	Ace Of Base				
6	Aerosmith				
7	Air				
8	Alan Parsons Project				
9	Alanis Morissette				
10	Alannah Myles				
11	Anouk				
12	Beck				
13	Bjork				
14	Black Crowes				
15	Blof				
16	Bob Dylan				
17	Bon Jovi				
18	Britney Spears				
19	Bruce Springsteen				
20	Bryan Adams				
21	Celine Dion				
22	Cher				
23	Chris Isaak				
24	Coldplay				
25	Daft Punk				
26	David Sylvian				
27	De Dijk				
28	Depeche Mode				
29	Destiny's Child				
30	Dido				
31	Dire Straits				
32	Doe Maar				
33	Duran Duran				

34	Enigma				
35	Eurythmics				
36	Fatboy Slim				
37	Frankie Goes To Hollywood				
38	Garbage				
39	Genesis				
40	George Michael				
41	Geri Halliwell				
42	Golden Earring				
43	Herman Brood				
44	Ilse Delange				
45	Inxs				
46	Jamiroquai				
47	Janet Jackson				
48	Jazzanova				
49	Jennifer Lopez				
50	Jewel				
51	John Hiatt				
52	Krang				
53	Kruder + Dorfmeister				
54	Kula Shaker				
55	Lauryn Hill				
56	Lenny Kravitz				
57	Madonna				
58	Marco Borsato				
59	Mariah Carey				
60	Mark Knopfler				
61	Marvin Gaye				
62	Massive Attack				
63	Meindert Talma En The Negroes				
64	Met Roosen				
65	Michael Jackson				
66	Moby				
67	Morcheeba				
68	Neil Young				
69	Nick Cave And The Bad Seeds				
70	Nirvana				
71	Oasis				
72	Pearl Jam				
73	Peter Gabriel				
74	Phil Collins				
75	Placebo				
76	Portishead				
77	Prince				

78	Prodigy				
79	Queen				
80	R.e.m.				
81	Radiohead				
82	Rage Against The Machine				
83	Red Hot Chili Peppers				
84	Rolling Stones				
85	Ronan Keating				
86	Ry Cooder				
87	Seal				
88	Shaggy				
89	Sinead O'connor				
90	Speedy J				
91	Spice Girls				
92	Spinvis				
93	Squarepusher				
94	Stereophonics				
95	Stevie Ray Vaughan				
96	Stevie Wonder				
97	Sting				
98	The Beatles				
99	The Cardigans				
100	The Cranberries				
101	The Cure				
102	The Eagles				
103	The Fugees				
104	The Orb				
105	The Police				
106	The Scene				
107	Toni Braxton				
108	Toploader				
109	U2				
110	Underworld				
111	Westlife				

F.2- Personal data

Name: .....

Age: .....

Gender: ❑ male
❑ female

Profession: .....

Do you use portable music players (i.e. walkman, mp3-player)?
.....

If yes, what devices do you use – and when do you use them?
.....
.....
.....
.....

If no, please explain why not?
.....
.....
.....
.....

Do you (often) play music from a collection in which you do not know many songs?
.....

If yes, how do you determine whether or not to play these songs?
.....
.....
.....
.....

F.3- General instructions

This experiment is about selecting songs on a portable music player, using two different methods. You will be asked to select a number of songs from a music collection containing 2248 songs from 111 different artists divided in 7 musical genres. You already received an overview of the artists in this collection. While selecting the songs, you can listen to the music at will.

The first method uses a standard list-based interface for song selection. With the second method, you can use a visual interface (the artist map) together with the list. The list is a subset of the complete system. You will get some time to practice with the system.

After using each method, a small questionnaire will be handed over. After you have used both methods, some questions will be asked about the system.

The songs that you select need to be different for the two methods. Both times, you are asked to select (and play a part of) 8-10 different songs. However, you should like the selected songs; they need to be *good according to your preference*. For helping selecting good songs, it may be wise to think about a particular situation for which you want to select the songs. This situation can be a ride in a car, music while walking through the park, commuting to and from work, etc. Note that since the test is for a *portable* music player, we would like the situation you think of to be *on the move* – and not at home where a full-fledged audio-system may be more appropriate.

Please, describe *your situation* for which you will select the songs:
.....
.....
.....

Good luck and thanks for your co-operation,
Rob.

F.4- System practice

The interactive system allows you to select music and listen to it. Either, you can select each song individually by its title, or you can be more vague in selecting the appropriate music. For example, instead of playing a specific song you can also play music from an album, artist, or even a genre in the list-based part of the system.

The artist map enhances the possibility to find music based on vague requirements, by enabling you to look for a certain kind of music without being explicit about what songs you want to hear. In the map, *tempo*, *mood*, *genre* and *year of release* are *attribute types* used to provide contextual information about the music on the device. Artists are represented by colored circles. The size, color, and position of a circle give information about the corresponding artist.

With the artist map you can select a subset of your music collection to investigate further (and play) in the list. By interacting with the artist map you control the context in the picture: you can change the attribute type used for coloring and the attribute type used for positioning. *Magnets* representing certain kinds of music determine the positioning of artists in the map; each magnet represents an attribute (i.e. *slow*, or *happy*). By changing the magnets, you change the *layout* of the artist map, because artists are attracted toward the magnets that describe their music.

To learn how to do all this using the system, we will guide you through a small step-by-step tutorial. You can ask the experiment leader questions and he will help you out. The system is controlled by a small touch-screen, to make the interaction with the interface more similar to interaction with a portable music player. About the interaction, i.e. the buttons:

> The play, previous and next buttons are for controlling music playback. The G, A, A and S buttons in the list stand for genre, artist, album and song and are used to move to the next or previous hierarchical level in the list. With the map button you go to the artist map. The named buttons at the top of the map interface change the active magnets. Magnets can be dragged, and clicking the coloring legend changes the attribute used for coloring. Artists themselves can be clicked, resulting in more information about the artist and a region for zooming. The + and - buttons are for zooming in and out. The list button is to go back to the list.

Try to perform the following set of small tasks as instructed.

- In the list, select Popular and Dance/Lounge as the genres from which you want to hear music.
- Navigate to the artists in these genres and select 'Britney Spears' and 'Madonna'.
- Press play to hear their songs in alphabetical order (skip songs at will), and navigate to the songs.
- Press pause to stop the playback, and select 3 songs from different albums by hand. Press play, and next to skip the rest of a song. What do you think happens if you press play when only a genre is selected? What if nothing is selected?
- Go to the map. In the map, select tempo and year magnets. Locate fast music from the 80s and zoom in on it. After zooming, notice the information on the bottom of the screen.
- Change the attribute used for coloring and the magnets used for positioning, and select some of the artists on the screen (one at a time) to get more information about them.
- Navigate back to the list. Notice that the shown artists are only the ones you zoomed in at.
- Select an artist that you know and like (if any), and one that you did not know before but think you may like given the information you got in the artist map (again, if any. If this does not apply to you, you can select another region to zoom in on). Play a part of some of their songs.
- Go back to the map, zoom out, and find out more about a kind of music you would like to hear.
- Play some music by either known or unknown artists that make music of this kind.

Now, play around with the system until you feel comfortable enough with the system to use it for real. When using the system for real, you can no longer ask the experiment leader for help.

F.5- Rules for each task

Try to select songs that fit the situation you have described.

The number of selected songs should be between 8 and 10, though not necessarily in one go (you may select 4 songs, listen to them, then select 3 other songs, and finally the last 3).

All songs in the selection should be different.

You are encouraged to add at least one song of an artist you did not know.

For the rest, you are free what songs should be selected, though the songs should reflect your preference as good as possible.

You are free to take as much time as necessary to select the songs.

When you can use the map, you should do so at least once; you should use the map to determine at least one artist from whom to add a song.

Start performing the task when the experiment leader signals you to do so.

Tell the experiment leader when you have completed the task (when you are satisfied with the selected songs)

F.6- TAM questionnaire

Please indicate to what extent you agree or *dis*agree with each of the following statements.

I find learning how to use the system easy.

strongly agree			neutral			strongly *dis*agree
1	2	3	4	5	6	7

I find it easy to get the system to do what I want it to do.

strongly agree			neutral			strongly *dis*agree
1	2	3	4	5	6	7

I find it easy to become skillful at using the system.

strongly agree			neutral			strongly *dis*agree
1	2	3	4	5	6	7

I find the system easy to use.

strongly agree			neutral			strongly *dis*agree
1	2	3	4	5	6	7

I find that by using the system I can easily select music that matches my state of mind.

strongly agree			neutral			strongly *dis*agree
1	2	3	4	5	6	7

I find that by using the system I am able to select music I want to hear rapidly.

strongly agree			neutral			strongly *dis*agree
1	2	3	4	5	6	7

I find that by using the system I enjoy selecting songs.

strongly agree			neutral			strongly *dis*agree
1	2	3	4	5	6	7

I find that the system helps me find unknown but enjoyable music.

strongly agree			neutral			strongly *dis*agree
1	2	3	4	5	6	7

I find the system useful for the situation I described.

strongly agree			neutral			strongly *dis*agree
1	2	3	4	5	6	7

I find the system useful for a portable device with a small touch-screen.

strongly agree			neutral			strongly *dis*agree
1	2	3	4	5	6	7

F.7- Preference questions

Which method do you like *most*?

- ❑ List-only
- ❑ List combined with artist map

Please motivate your choice?

.....
.....
.....

Which method do you like *most* for selecting music in the situation you described?

- ❑ List-only
- ❑ List combined with artist map
- ❑ The interface of my portable music player, namely: .....
- ❑ Something else, namely: .....

Please motivate your choice?

.....
.....
.....

For which method did you *like* the selected songs *better*?

- ❑ List-only
- ❑ List combined with artist map
- ❑ Both were equally well

Please motivate your choice?

.....
.....
.....

F.8- General remarks

What is the worst thing or component of the interactive system?

.....
.....
.....

What is the best thing or component of the interactive system?

.....
.....
.....

What should definitely be added to the interactive system?

.....
.....
.....