Requirements Document

The XMMS2 Client Project

XMMS2 is an open source project aimed at creating the next generation music player/manager. Currently, the project exists as a music database and playback server with an elaborate API for client applications. Essentially, it is a back end to a music player that handles looking up files, decoding various audio formats, and outputting sound under a veriety of system environments. There are several client programs for XMMS2 currently under development, but many of them are incomplete and unusuable, and none of them fully take advantage of the capabilities of the XMMS2 database.

The XMMS2 project was created because the current options in music management and playback software were considered limited and unsatisfactory by advanced users. The project was designed to be very powerful, but also abstract. It is meant as a starting point for a wide variety of music projects that offers much potential, but doesn't apply many limitations to the end product.

What this XMMS2 Client Project aims to create is a GUI client for XMMS2 that will improve on current music management software options in several key ways. This new project will mainly be useful for users with very large collections of music. It will make both finding individual songs and constructing playlists faster, easier, and more natural. It will do this primarily through use of Collections[1].

Goals

- Play music.
- Be simple enough to accomodate most computer users.
- Provide methods to quickly find songs in the user's library and organize them into playlists.
- Fully implement collections. (No current XMMS2 client does this)

[1]The XMMS2 Project defines collections merely as unordered subsets of a user's music library. Collections can be used to group related songs together arbitrarily and without any hierarchical limitations. New collections can be generated from the intersections and unions of other collections. This is the new key concept that the XMMS2 Developers hope will revolutionize how electronic music is managed.

Primary Features

- Play music.
- Browse the user's library.
- Search the user's library by ID3 tag information.
- Browse collections.
- Browse songs that are not yet in any collection.
- Quickly and easily add/remove songs from collections. (This has to be trivial to do and something the user can do casually while listening to the songs)
- Automatically generate collections from ID3 tag information and song meta data, such as last play date, play frequency, etc.
- Generate new collections from logical combinations of existing collections.
- Allow for logical collections, not just hand-picked collections. (Let collections be generated on the fly from a filter or from set operations instead of generating them statically from the current library/collections.)
- Construct and manage playlists by hand.
- Construct playlists from collections.

Advanced / Wish-List Features

- Allow for mixed logical and hand-picked collections. (Say you take two collections A and B and find their union to generate collection C. If you add song s_1 to collection C and s_2 to collection A, C should then contain both s_1 and s_2.)
- Maintain data about frequency and tempo to find similar songs by sound quality. (This could be further specialized to searching for songs that begin similarly to how another song ends, etc.)
- Search songs by lyrics.
- Use sound-pattern recognition to look up a user's collection online and correct eronious ID3 tag data (perhaps using MusicBrainz[2]).
- Share your collections with other users so they can categorize their music the same way you did without organizing their library by hand.

[2]http://musicbrainz.org/

Challenges

GUI Design

The main challenge of this project will be to create an intuitive and easy to use interface that allows for managing the currently playing songs, the full music library, many collections, and many playlists. That is a lot of diverse functionality that could easily get somewhat out of hand. It's important to keep the interface simple for basic interactions, and unintimidating for basic users.

Divisibility

Depending on how many of the advanced features are implemented, most of the program will likely be a graphical user interface. These interfaces are notorious for having complex webs of interconnection, and are therefore difficult to divide into smaller tasks. However, a good object-oriented GUI toolkit should allow each component to be developed separately, allowing division without too much pain.

Beyond the GUI, there are also several non-graphic features that could be written independent of the interface, especially if one or two of the advanced features are included in the final specifications.

Learning Curve

This project will require most of the people on it to learn how to use the XMMS2 API, and many will have to learn at least the basics of some GUI toolkit. Although using these libraries will make writing the project much easier on the whole, there will be a lot of learning involved, and that will take time.