Fractal Blasters Media Player Module Guide

This guide to the Fractal Blasters Media Player is modeled after the A-7E Module Guide.

Goals of the Module Structure

The Fractal Blasters Media Player module structure has been designed to provide separation and code guidelines. The module structure should:

1) Define the modules that will be implemented
2) Separate the work between developers, so there are not conflicts when code is committed into the repository
3) Keep code independent from other modules, so a developer is not concerned with other developers' implementation.
4) Show the relation between modules

Design Principle

The module structure has been divided into three categories: Wave Processing, User Interface, and File Management. All modules that are developed should be put into one of these categories. The Wave Processing category contains all modules that handle audio data. The User Interface category contains all modules that present a form to the display. The File Management category contains all modules that store file information in data structures.

Module Descriptions

This section defines each module by listing the service provided, and the secrets that are not visible to other modules.

1 Wave Processing

The wave processing category contains modules for Input, Effects, Output, and Playback Control. The services provided usually involve a stream of PCM audio data.

1.1 Input

Service
Provides a stream of PCM data decoded from an audio file.
Secret
How to open an audio file, decode the information, and return a PCM stream.

Associated variabilities and parameters of variation
V1. Supported file formats

1.2 Wave Plugins

Service

Receives PCM data from input and sends commands to any associated plugin form. May modify PCM stream before it reaches output.

Secret

What effects the PCM data may have on a plugin form, how the PCM data is modified before output.

Associated variabilities and parameters of variation

V4. Visualizer included

V5. Visualizer level

1.3 Output

Service

Receives PCM data and outputs the data to the Windows playback device.

Secret

How the data is passed to the Windows driver.

Associated variabilities and parameters of variation

None

1.4 Playback Control

Service

Controls wave processing modules based on commands received from playback control form.

Secret

What commands will be issued to wave processing modules when a certain command is received from the playback control form.

Associated variabilities and parameters of variation

None

2 User Interface

The user interface category contains modules that have visual elements, and controls that are accessible by the user.

2.1 Plugin Forms

Service

Provides user control of plugins. Displays information that a plugin provides.

Secret

How the form is drawn. How user events are forwarded to a plugin.

Associated variabilities and parameters of variation

V4. Visualizer included

V5. Visualizer level

2.2 Playback Control Form
Service
Provides user control of playback. Relays messages to playback control.
Secret
How the form is drawn. How user events are handled.

Associated variabilities and parameters of variation
None

2.3 Playlist Form
Service
Lists contents of a playlist. Provides user with ability to select specific song.
Secret
How the form is drawn. How user events are forwarded to the playlist module.

Associated variabilities and parameters of variation
V3. Saving/Loading playlists
V6. Multiple playlist support

2.4 Library Form
Service
Displays contents of media library. Provides user with ability to select specific artist.
Secret
How the form is drawn. How user events are forwarded to the library module.

Associated variabilities and parameters of variation
V2. Sorting music by category

3 File Management
The file management category contains data structures that store file paths and media information.

3.1 Library
Service
Stores file paths and media information that can be sorted, filtered, and searched.
Secret
The data structures used to store the information.

Associated variabilities and parameters of variation
V2. Sorting music by category

3.2 Playlist

Service

Keeps a list of queued files. Provides method to get current track, next track, previous track, or track by index.

Secret

The data structures used to store the information.

Associated variabilities and parameters of variation

V3. Saving/Loading playlists

V6. Multiple playlist support

Issues

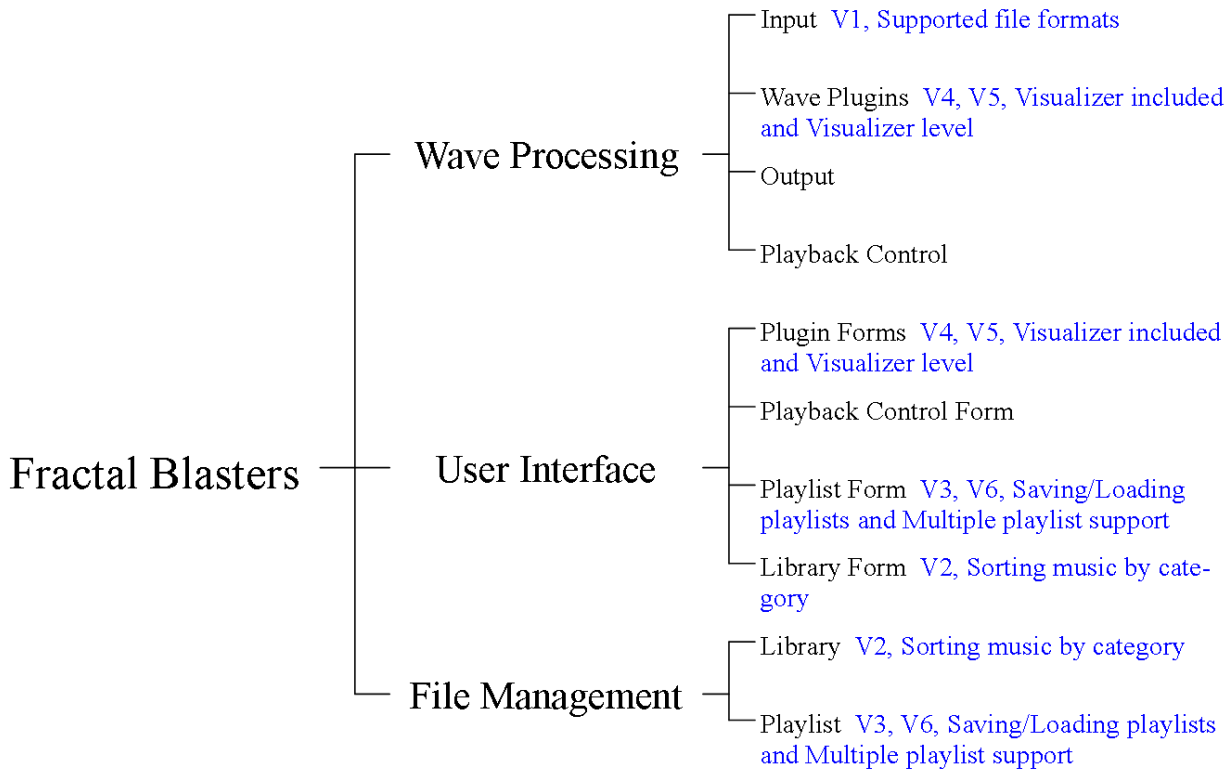
Issue 1: Should the playlist module be able to open files that are not part of the library module?

A1: Yes. The playlist module should keep references to file paths so that files can be played without adding them to the library.

A2: No. The playlist module should keep references to song objects that are stored in the library module. The library module will contain the file paths. If a user wants to play a new file, it will be first added to the library, then to the current playlist.

Resolution: TBD

Fractal Blasters Module Structure



Review Questions

1. Is there necessary functionality (see Commonality Analysis) for which it is not clear how such functionality will be supported in the current module structure? i.e. does every commonality map into one or more modules in this document?
   Yes
2. Does each variability in the Commonality Analysis map into one or more modules of the module guide?
   Yes
3. Are the mappings between variabilities and modules correct?
   Yes
4. Is there any variability that presently maps into too many modules? If so, how might this be fixed?
   No, the variabilities should be present in both the user interface modules and the functional modules.
5. Is there any module that presently hides too many variabilities? If so, how might this be fixed?
   No
6. For those modules that do not have any commonalities or variabilities mapping into them, why is this the case? Is the module necessary? Is the module used by other modules that provide functionality required by the Commonality Analysis? Does the module represent a requirement that was not captured in the Commonality Analysis? If so, should this requirement be in the Commonality Analysis?
   The modules Output, Playback Control, and Playback Control Form do not have variabilities. They will be the same in every release.
7. Are there any modules that should be added?
   No
8. Are there any modules that should be removed?
   No
9. Are there any modules that should be split?
   No
10. Are there any modules that should be merged?
    No
11. Is any functionality duplicated between modules? Does more than one module have the same secret?
    No
12. Are there any lower-level modules that should be in different higher-level module?
    No
13. Are there any modules that seem excessively difficult to implement?
    No
14. Is every leaf module of an appropriate size for implementation?
    Yes
15. Does every module have the correct description and secret?
    Yes
16. Does every module have a description and secret that is easily understood?
    Yes

17. Does the module structure, particularly when one looks at the "allowed to use" relationship in the accompanying document, support iterative development? Are modules relatively independent, such that they could be developed and tested incrementally, or do the dependencies between them require that signification testing could not be done until many modules were completed?
    Significant testing can be done for each module independently if simple test classes are made.
18. Is every issue convincingly resolved?
    Yes
19. Does this document need a dictionary? If so, is the dictionary from the Commonality Analysis sufficient, or is there additional terminology in need of explanation?
    No