System Generation and Verification Plan for Fractal Blaster

Do we need an introduction? Probably. **System Generation & Verification Plan for Fractal Blasters Media Player**

**Team Name: Fractal Blasters (9)**

**Authors: Marvin Toeung, David Koster, Greg Hanes, Keenan Lang, Cliff Braton**

**Test Plan**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Capability to be tested** | **Commonalities & Variabilities** | **Tester(s)** | **Preconditions** | **Expected Result** | **Test Case** | **Completion Date** |
| Product Line Generator | All | Marvin Toeung | All code compiles | No Delay Loaded DLL erorrs, Only selected features present | TC1 | April 1 |
| Audio Decoding | C1, V1 | David Koster | Decoder library present | PCM data representing the contents of the audio file | TC2 | April 1 |
| Audio Playback | C2, C4 | Marvin Toeung | Output library present | Sound from speakers corresponding to audio file | TC3 | April 1 |
| Metadata | C6 | Marvin Toeung | Metadata library present | Library prints all ID3 tag data for input fie | TC4 | April 1 |
| Playlist | V3, V6 | Marvin Toeung | Playlist library(s) present | Playlist window appears | TC5 | April 1 |
| Media Library | C6, V2 | Keenan Lang | Media library library present, Metadata works | Library appears, is sortable, allows selecting files to play | TC6 | April 1 |
| Audio Effects | NA | Marvin Toeung | Effects library(s) present | Audio played back is different according to the description of the plugin | TC7 | April 1 |
| UI | C2, C3 | Marvin Toeung | All base plugins present | Buttons control playback, menus operate properly | TC8 | April 1 |
| Visualization | V4 | Greg Hanes | Core functionality | Visual that changes with audio input data | TC9 | April 1 |
| Dynamic Plugin Loading | All | Cliff Braton | None | Functionality is added by placing plugins in proper location | TC10 | April 1 |
| Endurance | All | Marvin Toeung | All Prior Tests | Program functions for extended period of time without crashing | TC11 | May 1 |

**Test Cases**

# Test Case 1: Product Line Generator Test

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step | Description | Input Type/Value | Expected Results | Setup |
| 1 | Select Product Model to test (will repeat for each) | Each of the product models available | NA | Start “Product Generator” application |
| 2 | Generate Product | NA | Product with selected functionality |
| 3 | Perform TC2-TC11 | See individual test case | Product passes all test cases as expected for its feature set |

# Test Case 2: Audio Decoding

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step | Description | Input Type/Value | Expected Results | Setup |
| 1 | Load Audio File | Open any suitable audio file (mp3, m4a, etc) | Decoder initializes | Start core media player application |
| 2 | Play Audio File | Play Button Event | MemoryStream fills with PCM data corresponding to the audio in the file |
| 3 | Seek Audio File | Stop Button Event, Play Button Event | MemoryStream fills with the same data as Step 2 |

# Test Case 3: Audio Playback

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step | Description | Input Type/Value | Expected Results | Setup |
| 1 | Load Audio File | Open any suitable audio file (mp3, m4a, etc) | Output device initializes | Start core media player application |
| 2 | Play Audio File | Play Button Event | Audio comes out through default audio device |
| 3 | Seek Audio File | Stop Button Event, Play Button Event | The same audio plays back through the default audio device |
| 4 | Pause Audio File | Pause Button Event, Play Button Event | The audio playback stops, and resumes when play is pressed |  |
|  | | | | |

# Test Case 4: Metdata

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step | Description | Input Type/Value | Expected Results | Setup |
| 1 | Load Audio File | Open a file supported by the current product model (see TC1) | File loads and all metadata is displayed on screen (Metadata struct can also be checked) | Start core media player application |
| 2 | Repeat for each supported file type |  |  |  |

# Test Case 5: Playlist

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step | Description | Input Type/Value | Expected Results | Setup |
| 1 | Open Playlist Window |  | Playlist UI Window(s) display | Start core media player application |
| 2 | Add Media To Playlist | Selected support media types | Metadata from media appears in the playlist |
| 3 | Play Media from Playlist | Double click item in playlist/select and generate a Play Button Event | Selected item begins playing |
| 4 | Next/Previous | Next Song Event and Previous Song Event | Playlist switches playing song to previous/next song based on button event |
| 5 | Repeat/Shuffle | Series of Next Song Events OR time lapse | Playlist follows the flow dictated by selected options |  |

**Test Case 6: Media Library**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step | Description | Input Type/Value | Expected Results | Setup |
| 1 | Open Media Library | Open Media Library from plugins list | Media Library UI Window appears, begins scan for media files | Start core media player application |
| 2 | Media Library Populates |  | Library fills itself with data based on scanned media files |
| 3 | Build Playlist | Click Events / Drag Events | The selected songs are added to the Playlist Window |
| 4 | Media Library Load (with database in place) | Close Library, Reopen Library | Library does not scan for media files and still display the same content as in Step 2 |

**Test Case 7: Audio Effects**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step | Description | Input Type/Value | Expected Results | Setup |
| 1 | Load Audio File | Open any suitable audio file (mp3, m4a, etc) | Plugin does not throw exception | Start core media player application |
| 2 | Play Audio File | Play Button Event | Audio comes out through default audio device with the effect of this plugin added |

**Test Case 8: User Interface**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step | Description | Input Type/Value | Expected Results | Setup |
| 1 | Start main program | Double click main executable | Main program window opens | NA |
| 2 | Select each menu option in sequence | Single Click Event | Menu Option description is followed |
| 3 | Repeat Step 2 until all menu options have been tested |  |  |
| 4 | Press each button without loading any media files | Single Click Event | No exceptions occur |
| 5 | Repeat Step 4 until all buttons have been pressed |  |  |
| 6 | Load Audio File, Repeat Step 4 until all buttons have been pressed | Suitable audio file (see product model) | No exceptions occur |  |

**Test Case 9: Visualization**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step | Description | Input Type/Value | Expected Results | Setup |
| 1 | Open Visualization Window | Click Event to show UI window | Visualization window appears | Start core media player application |
| 2 | Load Audio File | Single Click Event | NA |
| 3 | Play Audio File | Play Button Event | A visualization related to the various qualities of the music appears |

**Test Case 10: Delay Loaded Dynamic Plugins**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step | Description | Input Type/Value | Expected Results | Setup |
| 1 | Start main project | Double Click Event on main executable | Main program starts | NA |
| 2 | Load a plugin using each type of interface supoort | Variety of click events | No exceptions regarding improper or missing DLLs |
| 3 | Ensure every plugin for the product model loads | Various | No exceptions regarding improper or missing DLLs |
| 4 | Ensure normal playback functionality occurs | Load a supported media file and play | Playback occurs properly |

**Test Case 11: Program Endurance Test**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step | Description | Input Type/Value | Expected Results | Setup |
| 1 | Perform each test case without closing the program | See each TC | All test cases succeed | Start core media player |
| 2 | Leave program running without media playing for 12 hours | NA | No excess memory consumed, program still running |
| 3 | Leave program running with media playing for 12 hours | NA | No excess memory consumed, program still running |
| 4 | Repeat Step 1 |  | All test cases succeed |
| 5 | Close Program |  | Program terminates without exception |  |