**Description**

1. The app will be a Yousician type design, supporting fingerpicking and chord playing.
2. Lyrics will be visible, probably under the fretboard display
3. The application will run in HTML5, Windows, Mac, Android thus it will need to be tested on a fairly regular basis in the unstable HTML5 port and bugs solved/reported as necessary.
4. The program will be for Ukulele, Dulcimer/Merlin, and theoretically, Diddley Bows, i.e. a variable number of strings, chromatic and diatonic fretting.
5. The input to the program will be a “compiled” form of music, not the raw text in the format **.music**
6. The compiled form of music will be generated by a single Python application which contains multiple compilers.
7. Source will reside in the tree with the final binaries, so no separate directories for source and object code.
8. Source music may be removed from the release version, as might copyrightable things (e.g. Ukulele Buddy)
9. Source will have consistent formats, e.g. the use of # for comments and := for equates.
10. Format 1 will be the “Big Book” format, known as **.strum** and will be the same format as the file “windows.strum” (When I’m cleaning Windows)
11. Format 2 will be the “Merlin book” format, e.g. a sequence on single fingerpicks known as **.tune**
12. The chord display will have to be redesigned to solve the HTML problem, hence a graphic fretboard with sprite dots.

**Format of .music files**

All lines have a prefix on the line which is nnnn.mmm: nnnn is the bar number and mmm is the position in that bar as a thousandth of a bar (e.g. 500 is half way through the bar). By convention all equate lines (the only thing that is not music/lyric orientated) are placed in bar 100 onward and numbered sequentially, and music actually starts at bar numbered 1000. Bar numbers do not have to be sequential and are created whenever something is created for that bar. The file must be ordered in bar:sub-bar order which is in practice an alphanumeric sort of the lines (which is easy to do with Python’s sort if it isn’t done anyway)

Format refers to the after the prefix (e.g. one would write 1004.000:”When I’m cleaning windows

With the exception of the *“lyric* format, spaces are removed and everything is converted to lower case.

|  |  |  |
| --- | --- | --- |
| **Format** | **Example** | **Notes** |
| #comment | # A song by me | A comment. Not normally used for anything but supported. |
| Key := Value | Tempo := 140 | Assign a value to a key. Both values are trimmed, the key is case insensitive. |
| “lyric | “Don’t worry | Sets the lyric for the *whole bar.*  This sets the lyric for the whole bar whenever it is used. Allows the use of % for expansion points. |
| <dir>[n,n,n,n] | u[0,2,3,2] | Play a string or a group of strings. The example is a Ukulele G chord. Strings can be “X” (don’t strum). Position 0 is an open string. <dir> can be u or d for up or down strum. |
| <dir>[@vv,n,n,n,n] | u[@42,0,2,3,2] | As above, but at 42% volume. |
| <chord> | <C#7> | Sets the display chord. This is purely display and can be anything at all, though individual chord names are used for colouring. An unnamed play is a fingerpick. The program correctly cases the display e.g. c7SUS will become C7sus |

Keys

|  |  |  |
| --- | --- | --- |
| **Key** | **Default** | **Notes** |
| tempo | 120 | Tempo of music in beats per minute |
| beats | 4 | Beats in each bar |
| instrument | Ukulele | Defines the instrument type, which defines other things (e.g. strings, tuning and so on) |