**Evaluation**

**Completion of the Objectives**

|  |  |
| --- | --- |
| Objective | Evaluation |
| A program should be written that takes text file containing a variant of the MML as an input, and outputs a single-channel MIDI file that can be play with conventional software. | I completed this objective fully |
| A program should be written that takes multiple MIDI files generated by the aforementioned program and combines them into a single multi-channel MIDI file. This combined MIDI file should be playable using conventional software also. | I completed this objective with the writing of the catmidi program. |
| The programs written for this project should:   * Complete their execution in under one second, so as to not interrupt the users work flow * Use less than half a megabyte of memory during execution * Be a less than a quarter of a megabyte in size | I met all of these objectives:   * Both programs complete their execution in well under one second, even when compiled without any optimisation. * While it depends on the size of the files given as inputs to the programs, both programs generally use well under 512 kilobytes of memory. * Combined, the programs take up less than 60 kilobytes of disk spaces |
| A version of the music macro language should be designed that will be used as the input for the program that generates a single-channel MIDI file from a single MML file. This language should:   * Have all the functionality of existing variants of the MML, including support for:   + Octave changing   + Accidentals   + Default length setting   + Volume changing   + Tempo setting   + Macros * Have as unambiguous a syntax as possible, with a clear logical progression | I completed this objective in my design section. The language I designed there exceeds the functionality of most MML versions with the inclusion of transposition. It also is necessarily unambiguous as a BNF grammar had to be written for it. |

**Evaluation of Development**

I believe that the development of the project went well – the use of Lex, Yacc and makefiles all proved to be good decisions, as each saved me a great deal of time. A area that was notably lacking during development however, was testing. I feel that I could have done more during the writing of my code.

**User Evaluation**

**What I Would Change**

If I were to do the project again I would implement a better system for testing my programs, though I am still unsure of how this would be achieved. I would also consider the idea of developing a single program to solve the problem, as having two programs to use has proved to be tedious at times. Finally, I would like to add more functionality to the music description language designed for the project, and perhaps make it more similar to a programming language: this could be achieved with the inclusion of loops, variables and control structures.

Fake user response