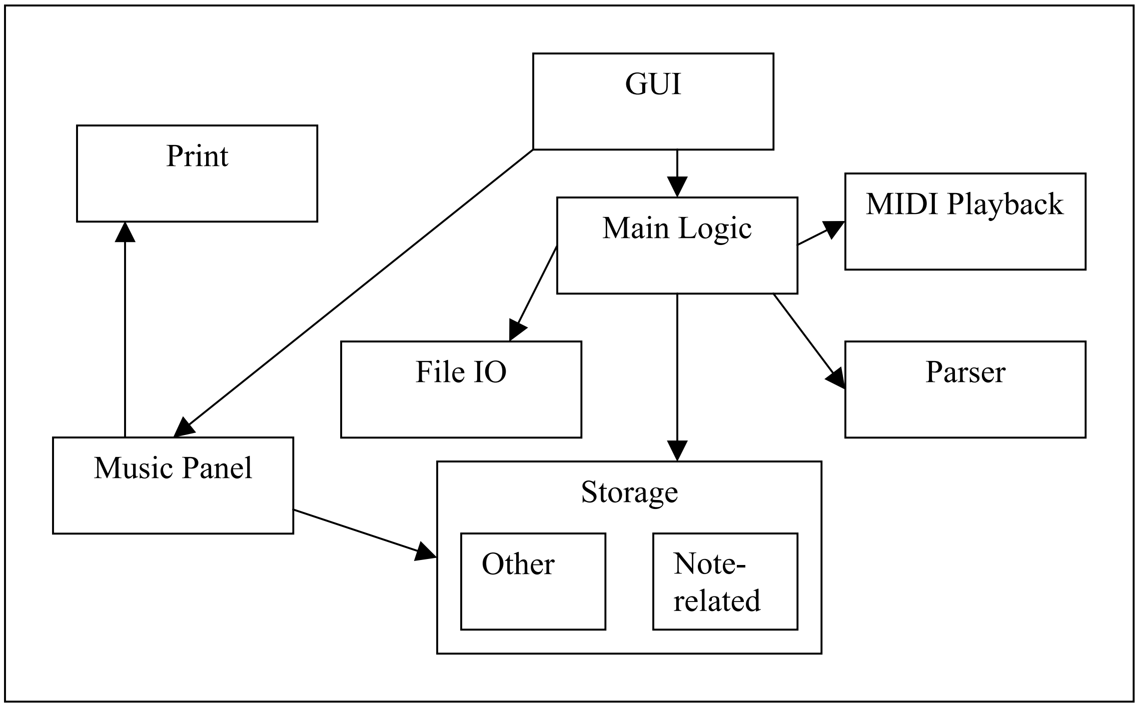
AMADEUS

Top-Level Design
by Kate Ho (siho)
20 February 2003

High-Level Component Diagram:



Description of Components:

GUI:

The graphical interface of the program. It consists of various menus and buttons for various functions (refer to specifications for supported functions). It takes user Inputs and passes it to "Main Logic". It contains "Music Panel".

Music Panel:

This is the interface for inputting, editing, and displaying music notes and related information (eg, key signature, meter, dynamics, repeats. See specifications for more details.) It has access to storage, which stores all these information. It contains "Print".

Print:

This component is responsible for formatting and printing the inputted musical composition. It should have "preview" and "print" function. It is strongly related to "Music Panel", as far as displaying music notes is concerned.

Main Logic:

This is the main logic of the whole program. It is responsible for passing information between different components, and controlling the flow of program. It is also responsible for all sorts of error checking, as well as undo and redo.

Storage:

This component stores all the information related to one composition. It uses measure as unit. It is responsible for transposing music. It is further divided into two components:

Note-related:

Positions of notes and rests, sharps, flags, note decorations, etc.

Other:

Key signatures, meters, beat/tempo, repeats, dynamics, special endings, lyrics, song info (title, author), anything not note-related.

Parser:

This component is responsible for parsing information stored in "storage" into midi format, and parsing midi formatted data to data in storage.

MIDI Playback:

This component is responsible for playing midi based on provided data. It uses external midi devices.

File IO:

This component is responsible for all file IO operation, including saving and loading data or any preference files.

External Dependencies:

This program depends on an external MIDI program for MIDI playback. It will also uses some well-developed graphic packages for drawing menus, buttons, and music panel.

Task Breakdown:

Project Leader and Structure Architect:

The person holding this role is responsible for the overall progress of this project. He/she is responsible for keeping everything on schedule, and keeping track what everyone is doing. He/she is also responsible for finalizing the interfaces between different components. Anyone who wants to make changes in connection with any other components must obtain approval from structure architect. This can be a task on its own, but if not enough coders are available, this position is best fit with "main logic."

User Interface Architect:

The user interface architect designs the user interface. Everyone could give his/her suggestions on UI to this person, but the UI architect has the final say on the designs. He/she is also responsible for drafting the user manual. This can be a task on its own, but if there is not enough coder, this position is best fit with "GUI." or "print".

Music Notation Researcher:

The music notation researcher researches on musical notations, and anyone with questions regarding this subject could refer to him/her. This position should be combined with coding one of the sub-parts of "storage".

Graphic Package Researcher:

The graphic package researcher researches about the graphical package being used, so anyone with questions could refer to him/her. This position should be combined with coding "GUI" or "print".

Storage Structure Architect:

The storage structure architect designs how data are being stored in system and file. Everyone could submit suggestions, but the architect has the final say. The person holding this role should code the main part of "storage", as well as either "file IO" or one of the sub-parts of "storage".

MIDI player researcher:

The MIDI player researcher researches about different external MIDI plug-ins and players, and decides which one would best fit to this project. This role should be combined with coding "MIDI playback".

MIDI format researcher:

The MIDI format researcher researches about MIDI formats. This role should be combined with coding "parser".

Testers:

The testers help testing components and different integrations. The persons coding "print", "MIDI playback", and "file IO" will probably hold this position. All group members are responsible for full system testing.

Tools developer:

The tool developer helps developing tools for this project. One useful tool at beginning could be writing a few dummy storages so other components could start before the "storage" component is fully implemented. This position is best fit with coding "file IO".

Documents Editor:

The document editor is responsible for collecting all the drafts from various architects and writes good user manuals and other documentations. It could be combined with any other job, so the person with least work load will probably get it.

Group Organization:

nige : project manager, main logic
dshue: main storage, other-storage subpart, storage structure architect
siho: note-related-storage subpart, music notation researcher, tester
mlsiegel: file IO, tester, tools
skim: Parser, MIDI format researcher, tester
tyoon: music panel
hyazawa: Print, Graphic package researcher, documents editor, tester
cphartne: GUI, UI architect
jkim: MIDI playback, MIDI player researcher, tester

Schedule:

3/9	Final Top-Level Design completed
3/12	Interface Proposals for each component completed
	Start researches related to component responsible
3/17	Receiving comments about interface proposals
	Start designing components
3/21	Final interface definition completed
	Researches completed
	Start coding
4/4	Detailed Designs submitted
4/12	Essential parts of components completed
	Component testing and debugging start
4/15	Integrations between pairs of components
4/16	Initial System Integration
4/18	All coding completed
	Massive component testing start
4/20	Component testing completed
4/22	Testing between pairs of components completed
	Full System testing and debugging start
4/27	Full System Implementation
	All documentation drafts completed
5/2	Public Demo
5/6	All documentation fully edited and completed
	Project should be ready to handed in
5/9	System submission

Assumptions:

The specification was not clear whether data are stored as a midi file or as a special formatted file, and so it was assumed that a non-midi specially formatted file would be used. It is also assumed that GUI would do the “Input Interpreter” work.