WAMS: Windowed Appraisal Movie System

This assignment will modify and extend LAMS. Almost all of the specified functionality of LAMS will be retained. In WAMS, the actions file has been eliminated in favor of interactions through the user interface that will be added.

Virtually all of the implementation for LAMS can be recycled into WAMS, provided the LAMS design is sufficiently good and the LAMS implementation is sufficiently complete. The major extension is that WAMS will incorporate a graphical user interface (GUI) implemented via the Amulet library. It is required that, during testing, all of WAMS's functionality be accessed via the GUI. WAMS will not use command-line arguments of any sort.

The only additions to the command set will involve the capability to edit a movie record to process a field change, the display of the very short synopsis/review of the film and simple searching will be provided.

Amulet GUI:

You must implement a GUI for this project. Extensive sample code showing how to use Amulet has been posted on the course website and discussed in class during the week of Oct. 30. Failure to have attended those discussions will most likely have a severely adverse effect on your ability to complete this assignment.

You have considerable latitude in your interface design. A prototype is illustrated below. All the specified functionality must be provided (for full credit), but you may vary the design in a number of ways if you like. In particular, you may implement buttons for some obvious operations, like sorting and navigation. You should NOT implement the entire interface in that manner.

You must implement at least four menus. The quality of your text labels in the interface is important, as is the layout of data in the windows.

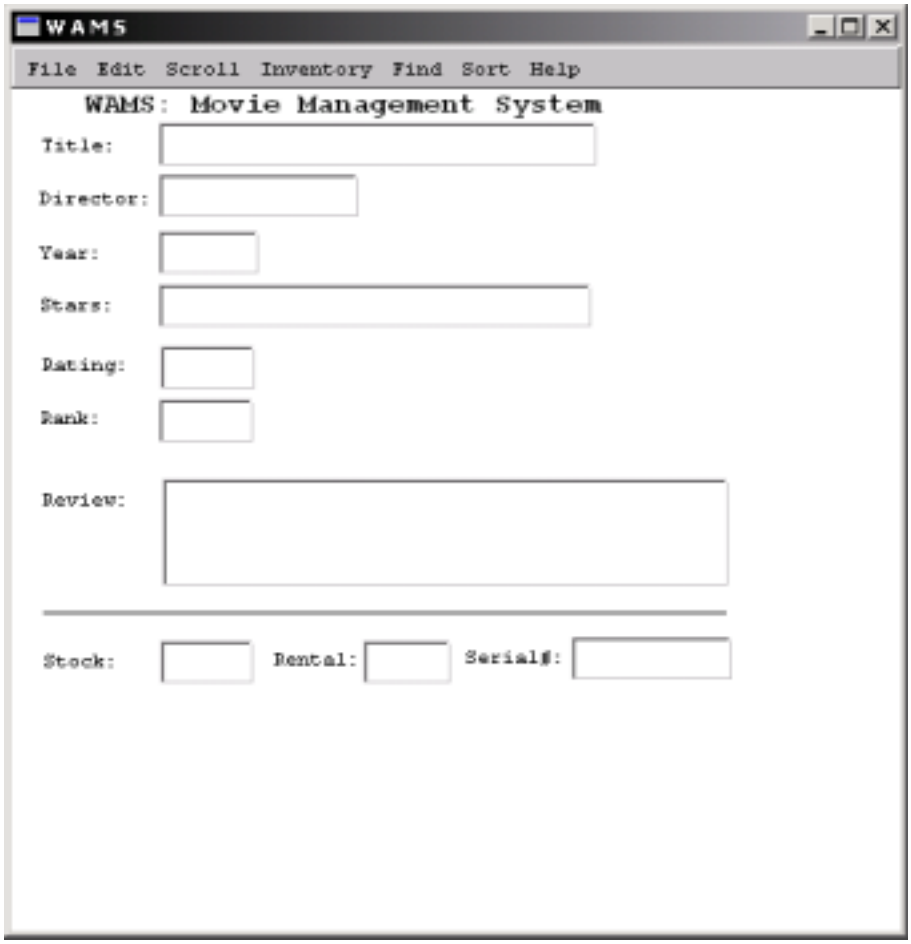
Prototype GUI:

Here's a possible main window for WAMS:

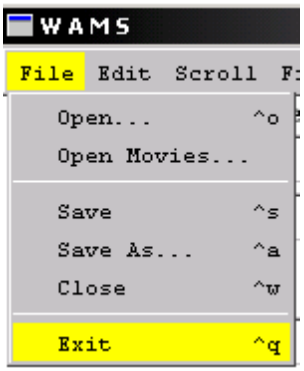
The client area shows labels for the display of information for a single movie at a time. On startup, no movie data has been loaded, so there is no data displayed.

The layout of the labels and accompanying data fields in the client area is up to you. All the fields shown here must be included. (Note that the "Stars:" and "Rank:" interface fields corresponds to the Performers and "Stars" database fields respectively.)

There are seven menus in the main WAMS window, each listing related choices.



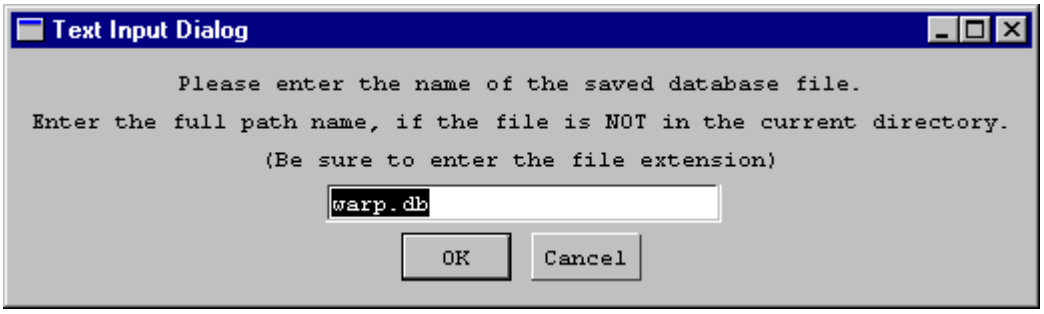
Here's the File menu:



There are choices for opening an existing database file, opening, (and converting) a movie data file, saving the current database list, saving the current database list to a file with a user-selected name, closing the current opened database file and exiting the program.

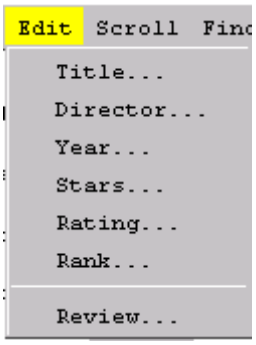
Note the "^q" on the Exit line. This is an accelerator (or keyboard shortcut). You may choose this menu item by holding down the Control key and pressing 'q'.

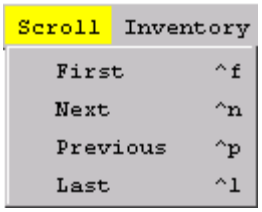
Selecting File/Open should raise a dialog box prompting the user to enter the name of a saved LAMS/WAMS database file, (note this corresponds to the Load command in LAMS), for example:



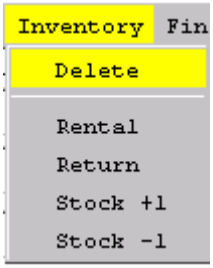
Once the user enters a file name and clicks OK, the file should be opened and read, and a new database list should be created, just as in LAMS. The window should be updated to display the first item in the database list:

The Edit menu is for future development purposes. In this beta development cycle for WAMS none of the Edit menu options will be functional. However, the user should be able to pull them down, but not select the options.

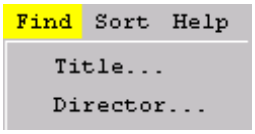




The Scroll menu should provide the ability to navigate the database list item-by-item in either direction and also to jump directly to either end, updating the window display accordingly. (This may be implemented as buttons if you want to learn more Amulet than is strictly required.)

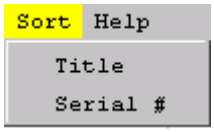


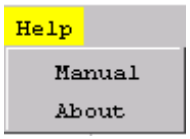
The Inventory menu will provide management operations for the movies. The commands operate upon the current displayed movie and will cause an update in the display. The Delete option removes the current displayed movie from the database. The Rental option signifies a check out of the movie, provided a copy is available. The Return option likewise option signifies a check in of the movie, provided a copy was checked out. The Stock options simply increment and decrement the stock total. (Stock cannot go below zero of course.)



The Find menu provides limited searching capabilities. Selection of an option will result in a popup dialog box for the user to enter a search string for the corresponding field. The search will be case sensitive, so the user must enter information for an exact match. WAMS will search through the current movie database attempting an exact match on the corresponding field. If a match is successful, the movie record of the matched string becomes the current movie in the display area. If a match is unsuccessful then no change to the display area occurs. In this case, it would be nice to give a message to the user but it is not required in this initial development cycle.

The Sort menu should provide choices for each field on which a sort is supported: Selecting one of the options should cause the (linked list) database list to be sorted on that key, and the window updated to show the first element in the newly sorted list:





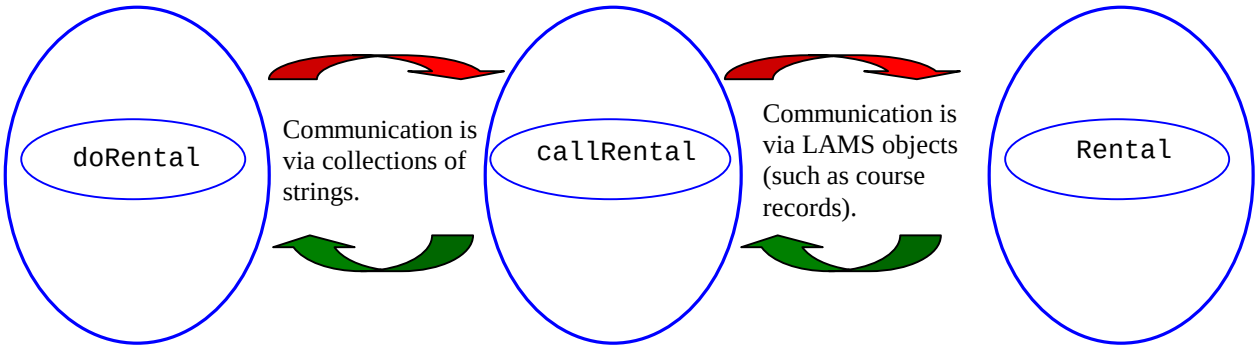
The Help menu contains one inactive option, the Manual option and one active option. The About option will display an startup/splash/about screen window giving information about the WAMS system and developer.

You should provide keyboard shortcuts for most, if not all, of the menu items. This is trivial in Amulet and will make it much easier to test your program. Warning: be very careful not to use the same shortcut for two different menu items.

There will no extra credit for creating exceptionally convoluted (or elegant) GUIs.

Integrating your GUI with LAMS:

The correct (and required) organization of your program has been discussed in class. The basic logical idea is that your program structure should be something like:



Amulet Code Translation Layer Code from LAMS

Input file descriptions:

The initial student data files have exactly the same syntax as for LAMS. The sample files posted on the web site for LAMS may be used to test WAMS, emulating the processing of the actions file via the graphical interface.

Output description and sample:

Output data resulting from a File/Save As command must be written to a file with the user-specified name. The format of this output is up to you, presumably you will use the same format as for the Save/Load commands in LAMS. There is no longer a dump command.

Linked List:

As with LAMS: you are **required** to use a linked list to store the student information. Your implementation of this list will be examined. In order to receive full credit, you must implement the nodes using a well-designed node class and the linked list itself must be encapsulated using a well-designed list class. In addition, the movie data records must be encapsulated (as a class) so that the data they store is logically separated from the node pointers.

Note this mimics the behavior of a C++ vector object. You are specifically forbidden to use any C++ vector objects in this program, or any other sort of predefined dynamic list type. You may base your implementation on the linked list implementations shown in the textbook, or those shown in the notes and lectures. You may not use linked list code from any other source. Violating that restriction would remove one of the major points of this assignment and will certainly result in a major deduction.

Programming Standards:

You'll be expected to observe good programming/documentation standards. All the requirements for documentation and coding given in the LAMS specification are still in effect. In addition your user interface code should be documented as well.

Deliverables:

There will be no interim design document for WAMS, since the design is essentially the same as for LAMS, aside from the Amulet components. The final design document should be in the same format as for LAMS, and should include the translation layer and the Amulet components.

Your final project submission must include the following (**and absolutely no other files**):

- all source code (`*.cpp` and `*.h` files) comprising your project
- a final design document reflecting the final design of your project, at the time of submission, including the Amulet components; this must either be in a format that can be viewed in MS Word or be a PDF file.
- a brief ASCII text readme file, named `readme.txt`, with any special execution instructions
- either the MS Visual C++ `.dsp` and `.dsw` files, or a UNIX makefile, as appropriate

As usual, you will submit your project to the Curator System as an archive file in one of two formats. For Windows users, submit a zipped archive containing the items listed above. (The program WinZip is very easy to use and is available from the University Computing Services website: `http://www.ucs.vt.edu/`) For UNIX users, submit a gzipped tar file containing the items listed above.

Note that omitting files from your archive is a classic error. Once you've created your project archive, copy the file to a new location, unzip it, attempt a build and then test the resulting executable. Submitting an incomplete copy of your project may delay its evaluation and **will** result in a substantial loss of points. In particular, if you omit a source file necessary to compile your program, you **will** be allowed to supply that file; however, we will then apply a late penalty corresponding to the date that you have provided a complete copy for evaluation.

Also note that including unnecessary files is also a classic error. Visual C++ users: do not zip up the **debug** subdirectory!

Evaluation:

WAMS will be subjected to runtime testing by the TAs, who will also score your implementation for adherence to the specified programming standards. Your score will depend on the quality of your design documentation, the quality of your coding and internal documentation, and the correctness and completeness of the functionality of your program during runtime testing.

Note that considerable weight WILL be given to the correctness of the functionality specified. It is therefore important that you fix as many problems in your LAMS implementation as you can. Simply providing a graphical user interface that does not accomplish much of anything will result in a poor score.

Amulet makes it very easy to disable menu choices. In the event that an option does not work, you should disable the choice but leave it on the menu. An acceptable alternative is to pop up a dialog box indicating that the corresponding feature is not functional at this time. It is much better to acknowledge these shortcomings openly than to knowingly run the risk that your program will crash if selection of a wrong action occurs during testing.

The input values entered during testing will be syntactically correct. You should still watch for logical errors. However, you may safely assume the entry of numerical data where it's expected.

You will sign up for a demo of WAMS, as you did for LAMS. Un-demoed projects will not be graded. Demos must be completed by 5:00 on Wednesday, Dec. 6. Since the TAs will also be taking exams, some TAs may require that demos be completed earlier than that, so sign up early.

Testing:

At minimum, you should test your program on **all** the posted input/output examples given for LAMS. Note that you can use the posted actions files to test the correctness of your implementation by entering the transactions manually through your GUI.

Pledge:

Each of your project submissions to the Curator System must be pledged to conform to the Honor Code requirements for this course. Specifically, you **must** include the following pledge statement in the header comment in the main source file:

```
//    On my honor:
//
//    - I have not discussed the C++ language code in my program with
//      anyone other than my instructor or the teaching assistants
//      assigned to this course.
//
//    - I have not used C++ language code obtained from another student,
//      or any other unauthorized source, either modified or unmodified.
//
//    - If any C++ language code or documentation used in my program
//      was obtained from another source, such as a text book or course
//      notes, that has been clearly noted with a proper citation in
//      the comments of my program.
//
```

Failure to include this pledge in a submission is a violation of the Honor Code.