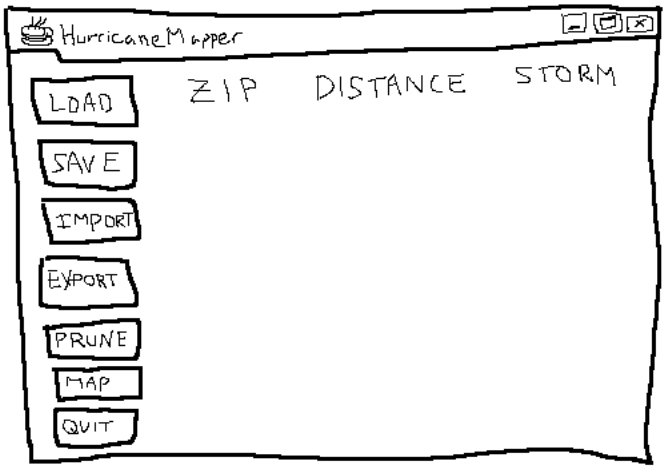
Project #3 GUI

Computer Science 2334

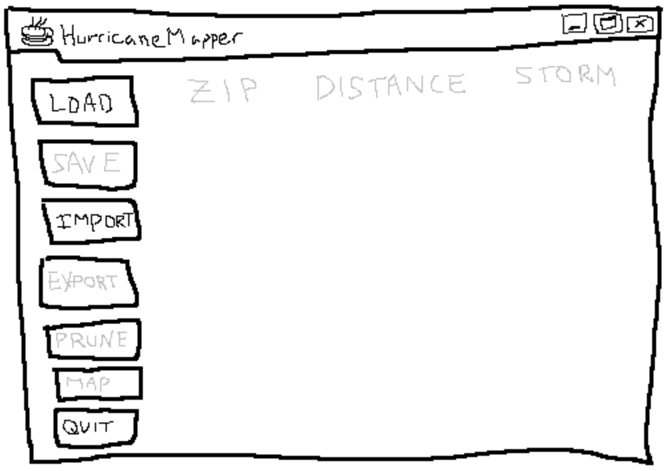
Fall 2007

Description:



The basic Graphical User Interface for Project #3 will have a main window with seven buttons down its left side and space for three lists in three columns to the right of the buttons. The seven buttons will be (in order) load, save, import, export, prune, map, and quit. The three columns will be zip, distance, and storm. A rough sketch of this is shown in the figure to the right. (The figure is intentionally rough so that you will not attempt to copy it exactly.)

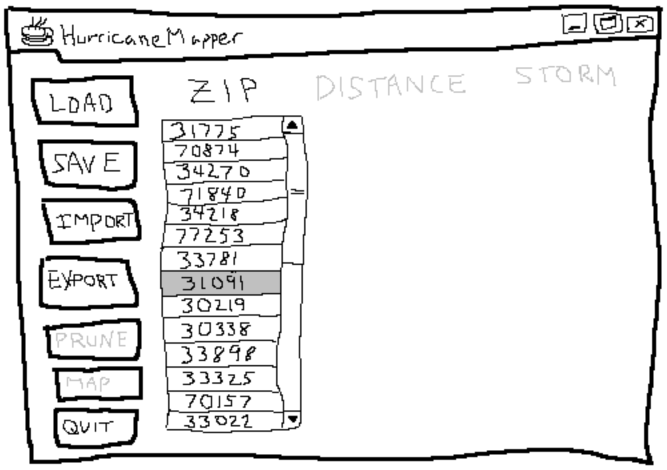
This main window will be augmented by additional windows that will pop up to help with file selection and mapping. In particular, a *JFileChooser* window will pop up to help with file selection for loading, saving, importing, or exporting data; and a ~~*MapPanel*~~ browser will pop up to display the resulting map. ~~*MapPanel* is a custom class that extends *JPanel* and includes a *JEditorPane* to actually display the map. *MapPanel* will be provided to you.~~ Code for creating JavaScript to allow the browser to display the resulting map will be provided to you.



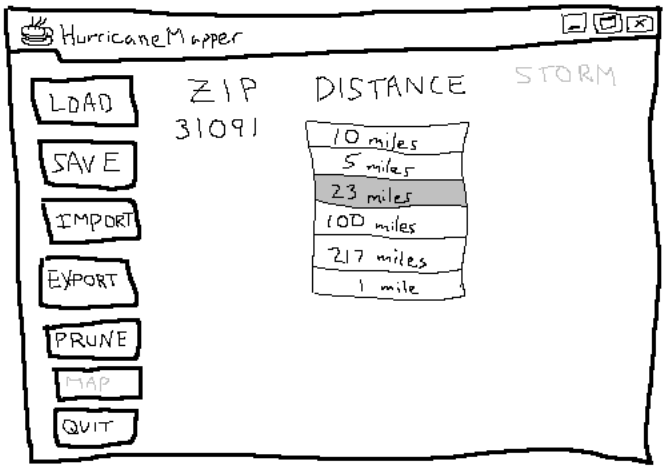
When your program is run initially, it should be clear from looking at the GUI that only load, import, and quit are possible. To make this clear, the save, export, prune, and map buttons will be displayed differently (e.g., with the letters "grayed out"). Likewise, until data is loaded or imported, the zip, distance, and storm columns will be similarly displayed in such a way that it should be clear there is nothing that can be done with respect to them. See, for example, the figure to the left.

When the load or import button is selected, a *JFileChooser* window will pop up to help with file selection.

Once data has been loaded or imported, the save and export buttons will become active. In addition, the zip column will become active. The contents of the zip column will depend on the data. If there is a single zip code present in the data, the contents of the zip column will simply be a display of that zip code. If there are multiple zip codes present in the data, the contents of the zip column will be a *JList* of all the zip codes in the data. With this *JList*, the user should be restricted to picking a single zip code. An example screen appears to the right.

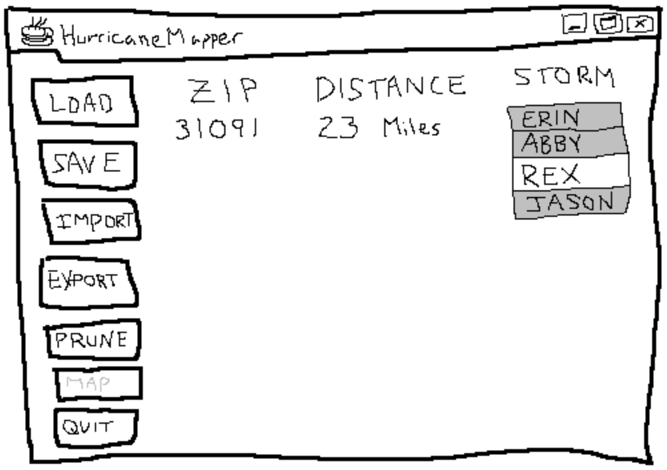


Once the user has selected a single zip code (or if there was originally just a single zip code in the data), the distance column will become active. As with the zip column, the contents of the distance column will be determined by the data. If there is a single distance for the zip selected, the contents of the distance column will simply be a display of that distance. If there are multiple distances present for the selected zip code, the contents of the distance column will be a *JList* of all the distances associated with the zip code in the data. With this *JList*, the user should be restricted to picking a single distance as shown below.



Note that, once the zip code has been selected, the prune button will become active (as well as the distance column). If the user selects the prune option at this time, all the data from the other zip codes will be discarded from your program's working memory. This means that if the user chooses to save or export the data at this point, only the data relevant to this one zip code will be saved or exported.

Once a distance has been selected by the user (or if there was a single distance associated with the selected zip code in the data), the storm column will become active. As with the previous two columns, the contents of the storm column will be determined by the data, with one data item simply being displayed and multiple data items requiring a *JList* for user selection. With this *JList*, the user should be allowed to select one **or more** storms as shown in the figure to the right.



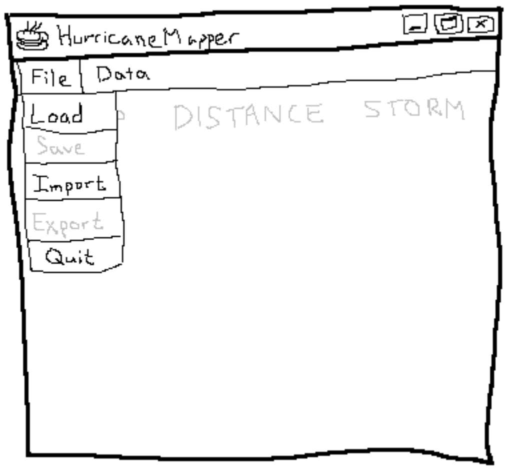
Note that the storm column will list storms by name. It is possible to have multiple storms with the same name. That is okay. List each **storm** one time, even if that means the same **name** is repeated multiple times.

Also note, if the user chooses to prune the data after selecting a distance value, then the data regarding the other distances will be discarded from your program's working memory.

Finally, once the user has selected one or more storms, the map button **may** become active. Whether it actually **will** become active is determined by whether there are 100 or fewer data points selected, including the locations of the zip code and all selected storms. (~~This is~~ While this was due to a limitation in the number of data points that Yahoo! will allow us to plot in a single map, and such a restriction does not appear to be present using the Google Maps API, we are going to retain this requirement in the program.) If there are 100 or fewer data points selected, the map button will become active. If there are more than 100 data points selected, your program will pop up a window to tell the user that it ~~can~~ will not plot more than 100 data points in a single map. Once the user has dismissed this popup window, he or she can either prune the data based on zip, distance, and storm, or select from among the previously chosen storms to further restrict the number of data points selected.

Extra Credit:

For 5 points of extra credit, you may replace the buttons from the description above with a menu bar. If you go this route, the menu bar will have a file menu with options to load, save, import, export, and quit, as shown in the figure to the right. It will also contain a data menu with options to prune and map. All the functionality of this GUI will be the same as the functionality of the button-based GUI described above.


HurricaneMapper
File
Data
Load
Save
Import
Export
Quit
DISTANCE
STORM