Video Game Sales

Piiashev Ivan 214-2

13.06.22

Julio Carrasquel

Problem statement

Creation of an GUI application which is able to modify, display and generate data, using Qt libraries and tools. During this process I have faced many difficulties, but at the same time learned to overcome them. I learned to structure my work, used number of new tools and approaches, understood how to work with visuals and now am able to create basic user interfaces.

Individual project specification

1. Subject area In some game store you were asked to make an application which will nicely manage sales data from more than 16,500 games stored in this CSV file. You are obliged to use C++ programming language with a Qt GUI based on Model-view internal design, implemented through Master-detail user interface pattern.

2.Data model The data is structured by the following attributes: “Rank”, “Name”, “Platform”, “Year”, “Genre”, “Publisher”, “NA_Sales”, “EU_Sales”, “JP_Sales”, “Other_Sales”, “Global_Sales”.

Attribute “Name” is Master in Master-Detail pattern (it represents entries).

3.Main features There are some things that you must implement: 1. Adjustability according to the screen size. Lay outing will help with it. 2. Proper structure. Your application must have main menu with two options for the user: “Start” and “About”.

A) About Clicking this button directs the user to the new popup window with following requirements:

1) A dialog window with information about the author in the form “Student No. ###”. Moreover, a logo must be implemented in this window too using Qt tools and overriding the paintEvent() method.

2) Option “Back” which closes the dialog window and in fact returns to the main menu. B) Start Clicking this button resizes main menu window to the new window with the following requirements: 1) Left side of the window is dedicated to the Master part (Names of the games). Pressing on one of the master’s entries releases a detailed view on the right side about this exact Master entry. 2) The user can add and delete entries of master part. By pressing the button “Add” your application creates a new window with all information which the user should fill about a game he or she adds. To delete some entry the user firstly must choose an entry, then press the button “Delete” which will delete this entry from master list. 3) The user can edit any information about a particular master’s entry.

4) The user can load a CSV file and save changes in it. By pressing the button “Save changes” your application must save all changes (obvious) and close the detail window. 5) UI elements that show number of games which updates respectively to changes (If user adds one more game, then obviously number increases by one) 6) A total value of sales by all fields which updates respectively to changes in exact master’s entry (If user changes EU_Sales in one game, then total sales changes too) 4. Additional features All these things relate to the “Start” section 1. You can make an ability for the user to sort Master list by some attribute 2. You can make an ability for the user to sort by several fields (for

example sort by year which is the prior sort and then by EU_Sales) 3. Item shows 4 fields of a game which has the most global sales (4 fields: Name of the game, Year, Genre, Publisher)

Details

Approach:

Project is centered around working with data. So, the main goal was to display the dataset. To do this I had to learn to work with Models and QStringList in my particular situation. Though I think table is more suitable for this purpose, as fields in my dataset were not containing lots of information. So, all the data would look ok in a table and also it would me more intuitive for sorting and filtering later on. I had to make a master list containing only one field of the dataset and to make it possible to access other fields when needed. To do this I created a button "Show details" which called Dialog window displaying all data. Secondly, I added another button calling another dialog but very similar to the Details one. It also contains of lineEdits corresponding to Game fields, but this time empty ones to be filled in by user. Right after that completed this one-by-one functionality with a Delete button. Then came a larger, harder task saving and loading files. Saving was easier as I remembered it from the seminar, loading took more time till I discovered it was a combination of all previous ones. I was adding rows in list like in AddWindow, Parsing the file like in the initial list creation, opening the file with a method similar to saving QString file = QFileDialog::getOpenFileName(*this*, "Open CSV file",("*.csv")); I was surprised it was filtering file types. I found out about after some internet research. Surprisingly for me, I was not doing this very often, most of the approaches were shown on the seminars. And my own difficulties were solved on a personal consultation. That was when I was creating About window and I didn't know how to override PaintEvent method. Another personal difficulty I faced were datatypes while parsing the file. Apparently, dataset provided for me contained not std::string types, but std::basic_string like array of chars. That caused me trouble when I was transforming strings into int and doubles. Stoi function didn't want to work as the datatype was wrong, so I changed to atoi and added .c_str() at the end. game.jp = atof(lineAsVector[8].c_str()); (double in this case). I also spent a lot of time trying to sort my model, tried a few different things and then found out It can just be made like this: *this* -> model->*sort*(0, Qt::*AscendingOrder*);.

This part can get very long, so here the link to the Git: https://github.com/pivandm/dsba-itop2022-hw

Discussion

As a person who has never programmed anything before the start of the university education, I found it difficult to keep up with the pace of learning. I thought that I would not be able to produce any code for the task, but as I think I covered many aspects. Its my biggest coding work, it took me a lot of time and effort to make it to this point. I obtained many important things that I missed during the year. Like syntax of the language, terminology and etc. I worked with multiple cpp and header files for the first time. At the end I got basic, but usable program that accomplishes most of the required tasks.

Conclusion

There is still a lot of to be done. I'm sure my code is untidy and maybe error prone or inelegant. I have done best to accomplish at least something. I also did all the work in 3 days. So, there is definitely room for improvement. At least to complete all tasks for 100%. Simultaneous sorting should be implemented, saving changes in Details, No 3. Aspect of additional tasks that I didn't understand also needs to be done.