

Antti Hannuksela (antti.hannuksela@lut.fi)

Otto Luttinen (otto.luttinen@student.saimia.fi)

Riku Pulliainen (riku.pulliainen@gmail.com)

WoWMobileArmory



The Idea

World of Warcraft Armory is character and item database of the game. It's main goal is to provide browser based user interface for in game data, like character/item stats, talents, in-game calendar, auction house and much more. Our goal was to make a simple front-end for QT-capable mobile phones, allowing users to browse character basic stats and item stats worn by characters.

Program usage is fairly simple. First you enter character name and realm, and then program fetches character and item data from Blizzards servers and displays it similarly as browser-based armory.

Because idea behind our application was already invented, our problems were only technical and making decision about what to implement, what kind of graphical user interface would be good and so on.

In character stats user interface we ended up using armory-like view, displaying items in three bars: left, right and bottom. Normally, in middle of the user interface would be 3d model of character, but as we haven't implemented this, we put character stats in middle. Item stat display is opened as popup by pressing icon of desired item.

Character selection interface is very straight-forwarded. It contains two text fields, one for character name and other for realm. For technical reasons, these fields are not on same page and user must press forward button between them.

Technically, program components consist of user interface components, downloader, download manager and XML parser.

Technical Specifications

There are actually two different layouts in the program. First layout consists of Begin button and two different QDialogs asking character name and realm. All those are just aligned to the top of the empty start screen.

Second layout is the main screen where character's stats are shown in textLabel which is positioned in the middle of the screen. Item slots are divided to three bars positioned in left side, right side and bottom of the screen. The quit button is located in the top of the screen. The alignment is done using QGridLayout. Item stats are shown when the icon of the item is clicked. QMessageBox is opened to the front of the current screen to show item stats.

Download manager is practically just map containing multiple downloaders and slots where downloader signals are connected. Managers task is to ensure that all download processes have finished before user interface components are drawn and that downloader objects are not killed before they have finished. Once everything required is downloaded, manager signals main program that user interface components are ready and exits.

Downloader is simple component, it gets address of page or file to be downloaded, sends HTTP request and saves reply(=requested file) to disk in cache folder.

XML Parser is component responsible of reading data from XML-files and storing it to QVector for other components to use. Parser is hard-coded to detect what data it is parsing by XML file name. Thus, parser knows what tags or attributes it must search and knows what vector parsed data goes. Parser uses SAX2 QT Api and QT XML libraries.

Program components have currently no tolerance on any errors whatsoever, so false input, broken XML file or failed file transfer will cause funny consequences. Also, application is designed to work when phone is rotated “upwards”, using wide screen will probably result in funny things in GUI.

Problems during development

Most of our problems during application development concerned GUI and object lifecycles. GUI was tricky to develop as it was completely new to our team and it had to be real-time editable because of button images. This one took most time of all technical problems we had. Another really tricky thing was lifecycles. The problem was, especially with downloader that caller objects finished before the downloader finished, resulting killing downloader too early. Sometimes we fought similar problem with GUI, when object called GUI creation and then excited, killing GUI. Some of these were fixed with object maps, some with better design and using signals/slots. Also Blizzards XML data was sometimes tricky to read, but these were quite easily overcome with if/else structures.

Because of lack of time the item stats component is still half-ready, meaning that it fetches stats of all items worn by character but fails to display them correctly. This issue is probably related to vector handling, as it seems that vector containing items stats is not correctly emptied after each stat display call.

Structure

The first thing the user will see is the begin button which starts the program. Next user will have to give character name and realm name. If either of these is wrong the program will start again and ask character name and realm name again. After the program has found the given character in the given realm it will go to the character screen which will show character stat in the middle and all the items the character is using in buttons with icons on the sides. There is also a quit button for stopping the program. In the character screen user can press a item button which will open a popup with the item stats. There is also a ok button that when pressed will bring back the character screen.

Conclusion

Because of the strict deadline the program wasn't completely finished. Some features were left out and some didn't work properly. One of the features that were left out was find a upgrade feature. With this feature the user would have been able to find out what items would be better than the item he or she is currently using and where to find them. Another feature that was dropped was the item browser. With this feature the user would have been able to find all the items his or her character could use or just browse all items.

All the features we did for the program worked pretty well considering the time we had except the item stats but that wasn't completely our fault. The xml file that contained the data for the items did contain a lot more waste than the xml file we got our character stats from and we didn't have the time to clean up the printing of the data. There was also another bug which was our fault. The program would print only the first items stats to the screen and if the user chooses another item after that the program would still print the first items stats to the screen.

Another bug which wasn't our fault was the order of the items buttons. This was the result of blizzards weird logic in numbering their items. In the xml file we got from the Internet every item was given a number but on blizzards website the items weren't in any kind of numerical order. So when we put them in a numerical order the items were in different places than in the website. This bug didn't stop the user from using the program. It was just something that would take some time to get used to.

All and all we were quite happy at the results of our work. We got all the main features implemented in to the program and most of them worked properly. The only thing that we wished we could have got to work was the item stats but because of the tight schedule we didn't have the time.