Technical University of Moldova

CIM Faculty

Report

Windows Programming
Laboratory work #1

Done by: student, gr. FAF-111

Name Surname

Verified by: assistant lecturer

MSc Truhin Alexandr

Chisinau 2014

Tasks and Points

Points	Task
0	-> create a Windows application
0	-> choose a *Programming Style Guideline*
0	-> create 2 buttons to window: one with default styles, one with custom styles (size, background, text color, font family, font size)
0	-> add 2 text elements to window: one with default styles, one with custom styles (size, background, text color, font family, font size)
0	-> add 2 text inputs to window: one with default styles, one with custom styles (size, background, text color, font family, font size)
1	-> Make elements to fit window on resize *(hint: you can limit minimal window width and height)*
1	-> Make elements to interact or change other element *(ex. on button click, change text element color or position)*
1	-> Change behavior of different window actions (at least 3). For ex.: on clicking close button, move window to a random location on display's working space **(1 pt)**
1	-> Write your own PSG (you can take existent one and modify it) and argue why it is better (for you) **(1 pt)**
4	Total

Replace this part with your contents (after you counted your points)

This template is ONLY FOR **RED PILL**. Paste in here text from formatted readme.

Easy way to do it is (I used MS Office):

1. Count your points (take in account Grading Policy)
2. Push changes to GitHub
3. Open README.md in browser
4. Copy file contents
5. Paste in here
6. Save
7. Export to PDF
8. Send

For example that's text from this page. Do not forget to add a link to your readme file.

First one to provide an automated method to generate such kind of reports will get 2-5 points. Fewer steps will have your solution, more points you'll get. An ideal solution may have 1 input box and 1 button in order to do everything.

Laboratory work #1

Window Handling

Purpose

Gain knowledge about basics of event-driven programming, understanding of window's class and basic possibilities of Win32 API. Also she will try to understand and process OS messages.

Mandatory Objectives

- Create a Windows application. It should have in middle following text: "Done with Pride and Prejudice by *student name*". Replace *student name* with your name.
- On windows resize, text should reflow and be in window's middle (vertically and horizontally)

Objectives with Points

- Limit minimal window width and height in order to keep text visible always **(1pt)**
- Change behavior of different window actions (at least 3). For ex.: on clicking close button, move window to a random location on display working space. **(1pt)**

Executed Tasks:

Mandatory Tasks:

- Create a Windows application
- Choose a *Programming Style Guideline* that you'll follow
- Add 2 buttons to window: one with default styles, one with custom styles (size, background, text color, font family, font size)
- Add 2 text inputs to window: one with default styles, one with custom styles (size, background, text color, font family, font size)
- Add 2 text elements to window: one with default styles, one with custom styles (size, background, text color, font family, font size)

Tasks With Points:

- Make elements to fit window on resize **(1 pt)** *(hint: you can limit minimal window width and height)*
- Make elements to interact or change other elements (1 pt for 2 different interactions) **(0-2 pt)** *(ex. on button click, change text element color or position)*
- Change behavior of different window actions (at least 3). For ex.: on clicking close button, move window to a random location on display's working space **(1 pt)**
- Write your own PSG (you can take existent one and modify it) and argue why it is better (for you) **(1 pt)**

Additional Tasks:

- Additional taks done for fun

Theoretical Work:

The programming language that I'll use will be C++. As I had to choose a PSG, I've chosen the USC Aerial Robotics Programming Style Guide (C++). I made some little changes that you can find in the [PSG_changes] file.

After struggling a bit with understanding the new concepts of window programming, I understood what is the role of the handle for an object. Also, I got interested by the idea of multiple messages in a queue and calling functions using these specific messages.

Practical Work:

Create A Windows application

I just followed the guideline from the WP tasks set. It worked.

Choose a Programming Style Guidline that you'll follow:

USC Aerial Robotics Programming Style Guide (C++). I made some little changes that you can find in the PSG_changesfile.

Add 2 buttons to window: one with default styles, one with custom styles (size, background, text color, font family, font size)

For the default button I used the system's default font. For the other one I createad another font:

```
FontCustom = CreateFont(16, 0, 0, 0, FW_DONTCARE, FALSE, FALSE, FALSE, FALSE,
                        0, 0, 0, 0, "Arial");
```

And there's also other button that I created another font specifically for it. The way I set the font, background and text color was by making the style of the button `BS_OWNERDRAW`. Afterwards I set the necessary properties of the button for the `WM_OWNERDRAW`.

Add 2 text inputs to window: one with default styles, one with custom styles (size, background, text color, font family, font size)

For the text input with default styles it was easy to set it up. I needed just to know how to create the edit and set it according to function parameters of `CreateWindowEx()`. The one with the custom style, was set by editing the case of message `WM_CTLCOLOREDIT` where I gave a background and text color. The font was set as for the button control.

Add 2 text elements to window: one with default styles, one with custom styles (size, background, text color, font family, font size)

The strategy with the control of text elements was similar with the ones for text input. There were some differentiations.

Make elements to fit window on resize (1 pt)

I made this by mentioning the case `WM_SIZE` and calling the `MoveWindow()` function. In the function I set for each control the behavior needed. Also, I set the minimal window size using the case of message `WM_GETMINMAXINFO`, where I set the sizes of the structure `GETMAXINFO`.

Make elements to interact or change other elements (1 pt for 2 different interactions) (0-2 pt) (ex. on button click, change text element color or position)

The first interaction is that the text elements were filled in with the text from the edit control, when the respective button was pressed (the left button for the left controls, and the right button, respectively). The second interaction is cleaning all the text controls when pressing the button CLEAR

Change behavior of different window actions (at least 3). For ex.: on clicking close button, move window to a random location on display's working space (1 pt)

I changed the behavior of all System Buttons of the window. The `WM_CLOSE`, when pressed gives a MessageBox, which informs that the user is taking the wrong way.
The `WM_MINIMIZE` actually maximizes the window in the whole system's display. And the `WM_MAXIMIZE`, gives a message in which it is asking the user weather he/she wants really to

close the app. If the user presses YES then the windows gets the message `WM_CLOSE`, which closes the application

Write your own PSG (you can take existent one and modify it) and argue why it is better for me.

The changes that I made to the USC Aerial Robotics Programming Style Guide (C++) are some simple ones. They do no So I described them here.

Conclusions:

During this laboratory work, I got more or less comfortable with event driven programming. I understood the deeper insight of how it is working. The way that the functions are used to change some features of the window was by sending messages to the main function and putting in the queue. Well, this was probably the most curious, because at first it was difficult for me to understand how does the Window Main Function does process all these messages, and that there are some messages that are not - queued.

An important remark is that each object, beginning with the window made, can not be created without a handle. Also, the fact that it is used as an identifier for the system to be used in functions.

Another thing that was interesting for me to learn is the thing that the controls like buttons, edit elements and text elements are considered themselves as smaller windows -- subwindows. For each type of these controls was a specific way to customize them and set a specific style. Another interesting thing are the system flags that are used to set some particular features to a control.

The fact that I had to use a PSG, made me enthusiastic about writing nice and compact code, using one of them. I cannot say though that the features and the guidelines made me perfectly comfortable with it, but I am still learning.