Technical University of Moldova

Faculty of Computers, Informatics and Microelectronics

Software Engeneering and Automatics Department

Event-Driven Programming

Laboratory work #1

Window. Window handling. Basic window's form elements

Author:
Vasile Drumea

Supervisor:
Mihai Coșleț

Chișinău 2018

Laboratory work #1

1 Purpose of the laboratory

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

2 Laboratory Work Requirements

- **Mandatory Objectives**
  - Choose a *Programming Style Guideline* that you'll follow
  - Create a **Windows application**
  - Add 2 `buttons` 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) *[one of them should be something funny]*
  - On windows resize, one of the `text`s should "reflow" and be in window's center (vertically and horizontally)
- **Objectives With Points:**
  - `(1pt)` Add 2 text inputs to window: one with default styles, one with custom styles (size, background, text color, font family, font size)
  - `(1pt)` Make elements to fit window on resize *(hint: you can limit minimal window width and height)*
  - `(0-2pt)`Make elements to interact or change other elements (`1pt` each different interactions) *(ex. on button click, change text element color or position)*
  - `(1pt)` 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
  - `(1pt)` Write your own PSG (you can take existent one and modify it) and argue why it is better (for you)

3 Laboratory work implementation

3.1 Tasks and Points

In this laboratory work i implemented all needed tasks. The application was implemented in CodeBlocks IDE in C++ language using the template for a Win32 GUI application. The program is as ussual with the 2 main methods : WinMain and WindowProcedure.

The WinMain method is almost as the default one except i changed the background color, name and position of window with the following snippets :

```
wincl.hbrBackground = (HBRUSH) CreateSolidBrush (RGB (100, 100, 100));
...
200,                 /* Windows decides the position */
200,                 /* where the window ends up on the screen */
544,                 /* The programs width */
375,                 /* and height in pixels */
...
_T ("WP Lab#1"),     /* Title Text */
```

In WindowProcedure which is the function called at every message i'm doing all the hard work. Initially i declare some variables which i'll use. I won't enumerate all because are many. In some words there are variables for fonts, for handles of controls, for brushes etc. The variables are named in such way to understand the purpose. Then with the main switch statement i begin to handle the messeges.

First message is WM_CREATE. In this case i create the required controls i.e. buttons and text fields. This are created with CreateWindow() or CreateWindowEx() for extended styles.

Then in case of WM_Paint i prepare the background and the used fonts. Also i send messages to the controls about the used font. E.g. :

```
SendMessage (button1, WM_SETFONT, (WPARAM) fontDefault, TRUE);
```

Also for the styling purpose i use the messages WM_CTLCOLOREDIT and

WM_CTLCOLORSTATIC which are used to change styles of edit and static text boxes. More precisely in those the brush is prepared and is sent to the control. Also there is specified the text color.

The positions of controls are specified in WM_SIZE. It is more comfortable this way because here we can make them change acording to the resizing.

The max and min size of the window are specified in WM_GETMINMAXINFO.

And now to add interaction between buttons and other controls we treat the WM_COMMAND message. there with another switch we add features for each button, e.g. :

```
...
```

```
case IDC_BUTTON2: {
SendMessage(edit2, WM_SETTEXT, (WPARAM) NULL, (LPARAM) NULL);
break;
}
...
```

The button Add is used to add the text form upper edit field to the other one and the clear button clears the second edit field.

And finnally to change the bahaviour of window actions we treat the WM_SYSCOMMAND. I don't think there is any explanation needed here. Just when minimizing we change the position of window to a random one, when maximizing change the bkgrnd color to a random one and for close button i included a warning dialog box before closing the execution.

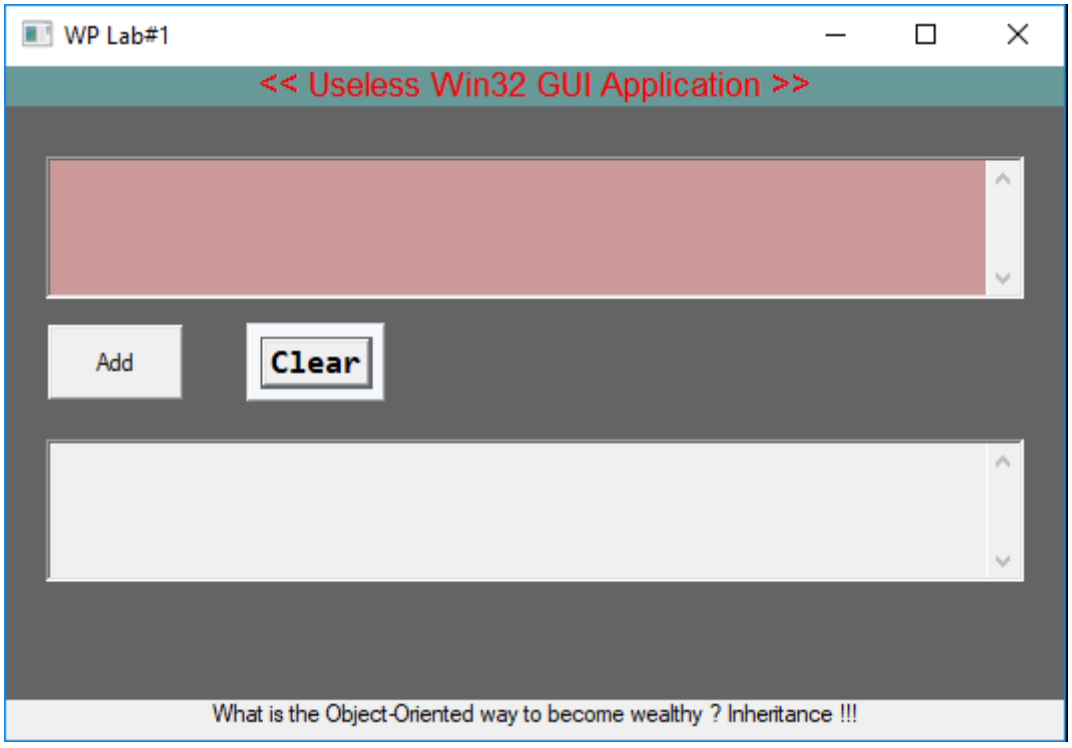
3.2 Laboratory work analysis

The analysis of all features was made in the above section. In the repository i included the .cbp file and the main.cpp file neccessary to build the app. Here is a link to my repository for Event-Driven Programming laboratories :

https://github.com/Wazea/WP

There is also my personal PSG. I chose to make my own PSG because i'm not comfortable with any other standart. I use a little bit of C-Style but also with elements from another languages like Java. I think that it is a good PSG because it provides good readability and a nice and organized look for the code.

3.3 Prove your work with screens



Conclusion

- In this laboratory work i gained knowledge about Win32 app developement. The concept of window, message handling and the way this proccess is organized.
- The execution shows that the program works as expected and the tasks were completed.
- The main idea of programming windows apps is based on getting messages and handling them(also sending them is needed).
- A good PSG is crucial in app developement to offer readabillity and to maintain the order in the code.
- It is recommended to include the instructions in the case acording to the message with which they are related. It is a bad practise to use same messages in different message handler because it makes the Window Procedure more chaotic.
- It is possible to send artificially a message with SendMessage().

References

1 MSDN, *Official page*, https://msdn.microsoft.com/en-us/library/windows/desktop/ff657751(v=vs.85).aspx

2 The First Reading, *Landing page*, http://www.winprog.org/tutorial/start.html