Faculty of Computers, Informatics and Microelectronics

Technical University of Moldova

WINDOWS PROGRAMMING

Laboratory work #1

Window. Basic window's form elements

Authors:
Daniel Surdu

Supervisor:
Irina Cojanu

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 she will try to understand and process OS messages.

2 Laboratory Work Requirements

– **Basic Level (grade 5 - 6) you should be able to:**

a) Create a Windows application

b) In the middle of the window should be present the following text: "Done with Pride and Prejudice by student name". Replace student name with your name.

c) On windows resize, text should reflow and be in window's middle (vertically and horizontally)

– **Normal Level (grade 7 - 8) you should be able to:**

a) Realize the tasks from *Basic Level*.

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

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

– **Advanced Level (grade 9 - 10) you should be able to:**

a) Realize the tasks from *Normal Level*.

b) Make elements to interact or change other elements (2 different interactions) (ex. on button click, change text element color or position)

c) 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

3 Laboratory work implementation

3.1 Tasks and Points

- Put text in the middle of the window

For this, I used the DrawText() function inside the WM_PAINT message in the following way:

```
DrawText (hDC, TEXT ("Done with Pride and Prejudice by Unnemotional Hyena"), -1, &rect,
DT_SINGLELINE | DT_CENTER | DT_VCENTER);
```

This function displays the text "Done with Pride and Prejudice by Unnemotional Hyena" in the center of the client area by using the DT_CENTER and DT_VCENTER identifiers. The first argument is a handle to the device context returned from BeginPaint(). The second argument is the text to draw, the third argument is set to -1 to indicate that the text string is terminated with a zero character and the forth is a pointer to the client rectangle. Also, on window resize it will remain in the middle because or window class style I used the CS_HREDRAW, CS_VREDRAW flags and DT_SINGLELINE from the name is clear that the text I wrote will be shown on the screen in a single line.

- Add buttons: with default styles and with custom styles

In order to create a button with default styles, the function CreateWindowEx() with the second parameter set to "BUTTON" was used. For creating one with custom styles we need to do the following. Firstly, a BS_OWNERDRAW flag should be introduced in the function. Secondly, in the WM_DRAWITEM message we have to draw this button from the beginning. If we'll not do that step we will get just a grey rectangle. In case WM_DRAWITEM I used GetTexExtendPoints32() funtion to get the text size, after that I set the text color using SetTextColor(), SetBkColor() to set the background color of the button, ExtTextOut() to write the text on the button and DrawEdge() to draw the edge of the button. To change the font of the OWNERDRAW button I used the SendMessage() function.

My buttons are placed below the text in the center of the main window, move simultaneously with the text and remain in the center of client zone when you resize the window. To make this I used the function MoveWindow() in the WM_SIZE case.

```
MoveWindow (button_2, x-20, y, 100, 40, TRUE);
```

- Add text elements: with default styles and with custom styles

In the same way as the buttons, the text element is created using the CreateWindowEx() function with the second parameter "EDIT". In order to change the text font I defined a custom font using CreateFont() and sent it with SendMessage(). Furthermore, I changed the text color and background color using the SetTextColor() and SetBkColor() in the WM_CTLCOLOREDIT message.

- Make elements interact or change other elements

When the "O O" button is clicked in the first TEXT("EDIT") appears the message: "You clicked the button "O O". This is made in case IDC_BUTTON_1 using funtion SetWinowText().

When the "HELL" button is clicked THE HELL BEGINS!!!, nope. When this button is clicked in the second TEXT("EDIT") changes the font randomly size and font style using CreateFont() function and SendMessage() function.

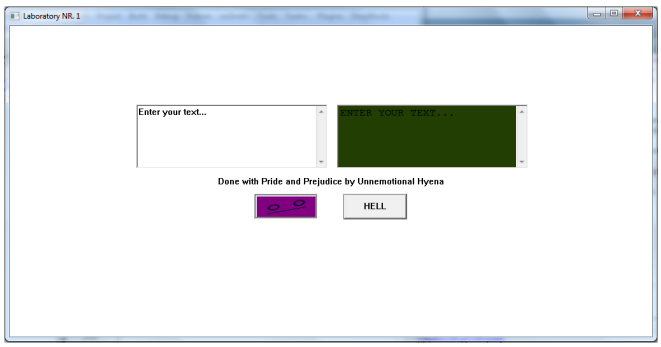
- Change behavior of different window actions

I changed the behavior of the three standard right top buttons and when I click in the window I get the message where is written the address of the program in the computer. In the WM_SYSCOMMAND I considered three cases: SC_MINIMIZE, SC_MAXIMIZE, SC_CLOSE. When minimize button is clicked I set the color of the background of the main window randomly using SetClassLong() function and generating randomly a RGB collor. When Maximize button is clicked I move main window using MoveWindow() at a random position from the left up corner to the coordinates of x and y = 400. When Close button is clicked appears a message box that asks you "Are you sure you want to quit?" and if you press "NO" it closes the window.

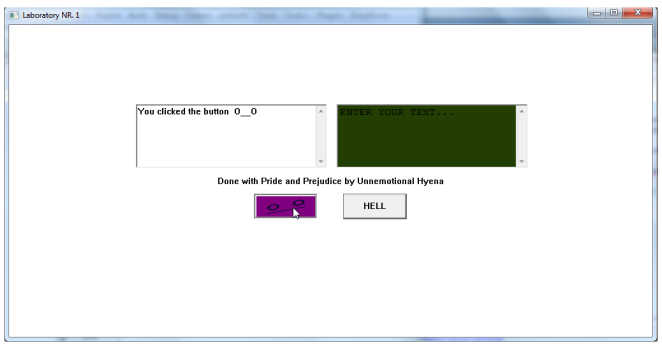
3.2 Laboratory work analysis

https://github.com/UnnemotionalHyena/WP

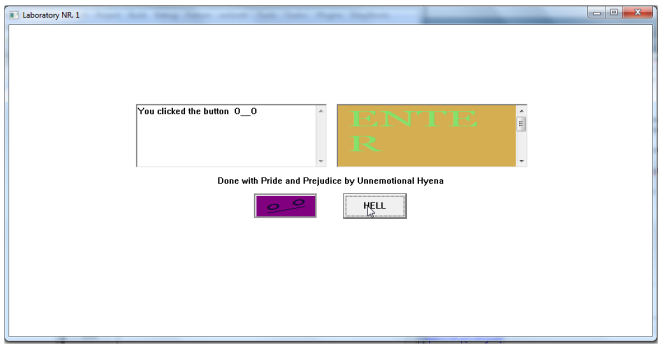
3.3 Prove your work with screens



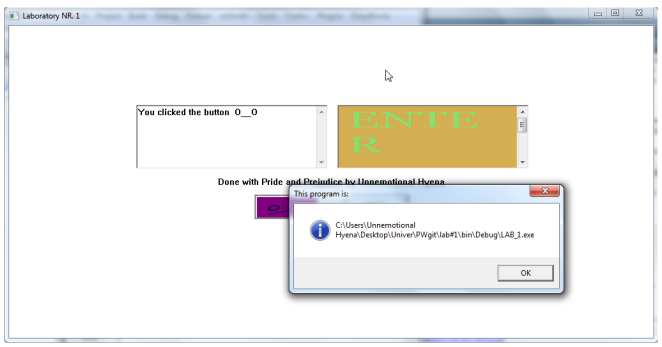
The basic window



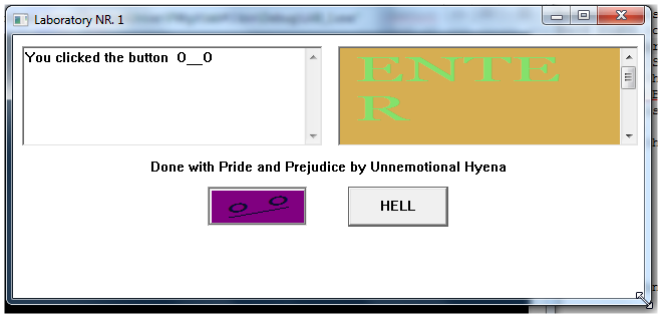
Clicking the O O button



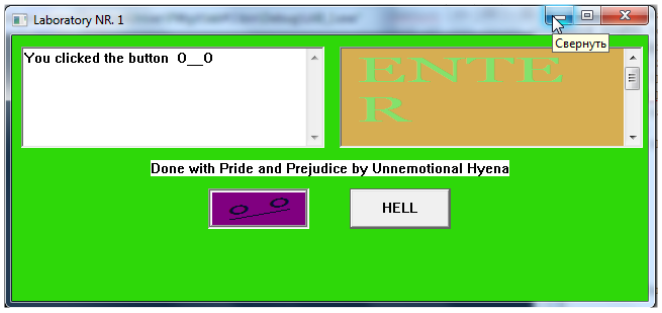
Clicking the HELL button



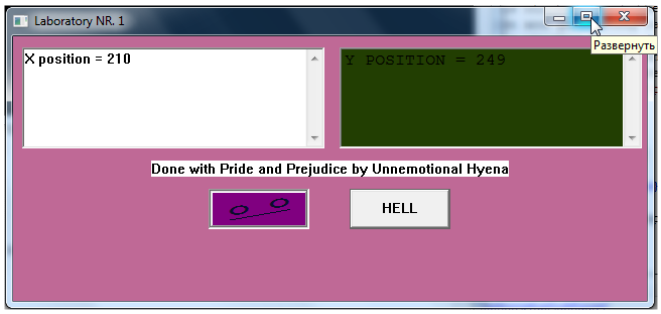
Clicking somewhere in the main window



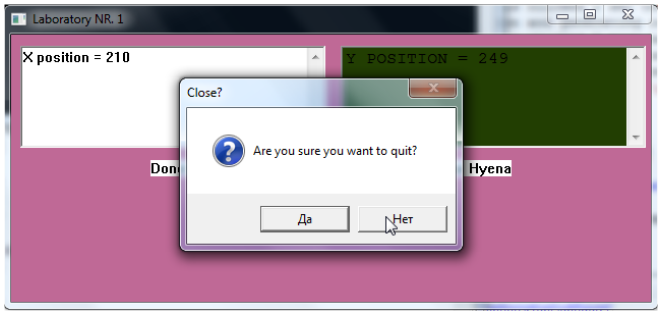
Resizing the window



Clicking the minimize button



Clicking on the maximize button



Clicking the close button

Conclusions

During this laboratory work I learned the basic of Windows Programming. I created a window that have two buttons and tow TEXT("EDIT") region one with the default style and one with OWNERDRAW style. This I made using C++ and windows.h library and it's functions. After that laboratory work I understood the event driven programming paradigm and got the basic idea of window programming.

References

1 Charles Petzold, *Programming Windows, Fifth Edition*, 1998

2 MSDN, `https://msdn.microsoft.com/`

3 Github, `https://github.com/`