

TECHNICAL UNIVERSITY OF MOLDOVA  
FACULTY OF COMPUTERS, INFORMATICS AND MICROELECTRONICS  
SOFTWARE ENGINEERING AND AUTOMATICS DEPARTMENT

## EVENT-DRIVEN PROGRAMMING

LABORATORY WORK #4

---

# Windows Timer. Animation.

---

*Author:*

Vasile Drumea

*Supervisor:*

Mihai Coşleţ

CHIŞINĂU 2018

## Laboratory work #4

### 1 Purpose of the laboratory

Gain knowledge about basics of working timer, its usage and how to implement animations.

### 2 Laboratory Work Requirements

#### – Mandatory Objectives

- Create an animation based on Windows timer which involves at least 5 different drawn objects;

#### – Objectives With Points:

- (2 pt) Increase and decrease animation speed using mouse wheel;
- (2 pt) Solve flickering problem, please describe in your readme/report how you did it;
- Add animated objects which interact with each other (2-6 pt), ex.:
  - \* Few balls which have different velocity and moving angles. In order to get max points, add balls with mouse, make balls to change color on interaction and any other things that will show your engineering spirit;
  - \* Any other interesting and reach in animation application;
- Animate a Nyan Cat that leaves a rainbow tail ( $\text{Math.floor}(+35\%$  for task with interacting objects))

### 3 Laboratory work implementation

#### 3.1 Tasks and Points

- (1 pt) Draw a custom bitmap image - Figure 3.4;
- (2 pt) Add a switch (button, select list...) that will change mouse ability to draw objects - Multiple tool buttons, 3 colors to choose, weight edit box and fill radiobutton;
- (1 pt) Draw a Bezier curve using mouse - Bezier spline for n points;
- (1 pt) Fill an object with a gradient - Fill radiobutton;
- (0 pt) Delete objects using mouse clicking - X;
- (2 pt) Use mouse as an eraser of adjustable width - Adjust its width with editbox;
- (0 pt) Zoom in and out application working area using keyboard - X;
- Total - 7 pt;

#### 3.2 Laboratory work analysis

The first mandatory task was implemented by creating different HPEN objects with the `CreatePen()` function. Then we construct the line with the specific methods e.g. :

```
static HPEN hPen1;  
...  
hPen1 = CreatePen (PS_SOLID, 1, RGB (255, 0, 0));  
...  
SelectObject (hdc, hPen1);  
MoveToEx (hdc, 100, 100, NULL);  
LineTo (hdc, 200, 200);
```

The result can be seen in Figure 3.1 in screens section.

For the second task I used `PolyBezier()` method and an array with 4 elements to draw a spline. The curvatures are controlled by mouse buttons. An example can be seen in Figure 3.2.

For the third mandatory and fourth task I implemented the drawing with the mouse of different objects. These objects can be of different colors and filled or not. Some examples are in Figure 3.3.

The tasks with points were implemented in the following way. I drew an example of a bitmap image with the available tools. It is represented in Figure 3.4.

For the tools I've created separate buttons. These are pushlike radiobuttons. Also there are 3 radiobuttons for each color. 3 colors are available in my application.

By default the pen tool and red color are checked.

```
Button_SetCheck ( GetDlgItem (hwnd, IDC_BUTTONRED) , BST_CHECKED);  
Button_SetCheck ( GetDlgItem (hwnd, IDC_BUTTONPEN) , BST_CHECKED);
```

To add functionality to the tools I treated 3 messages: WM\_LBUTTONDOWN, WM\_LBUTTONUP, WM\_MOUSEMOVE. In the first and third, depending on the color and tool I create the pen, brush and draw what is needed. There is an issue that i could not solve. Because I use foreground mix mode R2\_NOTXORPEN when the figures overlap the obtained color is a combination of all colors and is not the last applied color as I desired to be.

To change the width of the line write the desired one in the edit box and click OK.

I thought that this can be solved if we would store in a stack all views after all changes, but though I thought that it is too complex and memory consuming for this lab work.

For the figures i.e. rectangles and ellipses there is the fill option to draw filled figures.

The eraser was implemented as a white pen, nothing special. It's width can be changed in the editbox.

The bezier spline is drawn with PolyBezier() function. You need to click as many points as you need for the spline. Actually it will work for 4 points and 7. After you click 4 or 7 points on working space rightclick and the spline will be drawn. It is specific for the function so it cannot draw a spline with an arbitrary number of points.

The part with the zoom I was thinking to implement it by selecting a part of the client zone and display that part, but there is the detail that we should maintain the buttons also in the view so i could not find a way to zoom only the working area.

In the repository i included the .cbp file and the main.cpp file necessary to build the app. Also there are 2 header files. The "macros.h" is for macros i.e. IDs and "declarations.h" is for the declarations of used methods. Also there is the cpp file with the methods, the rc file with the icon and the .ico file.

Here is a link to my repository for Event-Driven Programming laboratories :

<https://github.com/Wazea/WP/tree/master/lab-3>

### **3.3 Prove your work with screens**

## Conclusion

- In this laboratory work i gained knowledge about Win32 app developement. The workflow with mouse and some GDI functions.
- The execution shows that the program works as expected and the tasks were completed.
- The main idea of an app like Windows Paint is to create different controls for tools and treat them at the mouse events.
- 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.

## References

- 1 MSDN, *Official page*, [https://msdn.microsoft.com/en-us/library/windows/desktop/ff657751\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/ff657751(v=vs.85).aspx)
- 2 Programming Windows by Charlez Petzold, 5th edition, SectionI,Chapter8