Technical University of Moldova

Faculty of Computers, Informatics and Microelectronics

Software Engeneering and Automatics Department

Event-Driven Programming

Laboratory work #3

Basics of Working with Mouse. GDI Primitives. Bezier Curve.

Author:
Dumitru Chetrusca

Supervisor:
Mihai Coșleț

Chișinău 2018

Laboratory work #3

1 Purpose of the laboratory

Gain knowledge about basics of working with mouse, GDI primitives and Bezier curve.

2 Laboratory Work Requirements

- **Mandatory Objectives**
  - Draw few lines of different colors and weights;
  - Draw a Bezier curve;
  - Draw few plane objects (ex. circle, square, pie, polygon...) of different colors, weights, filled and not;
  - Draw 2 different objects using mouse
- **Objectives With Points:**
  - Draw a custom bitmap image; (1 pt)
  - Add a switch (button, select list...) that will change mouse ability to draw objects; (2 pt)
  - Draw a Bezier curve using mouse; (1 pt)
  - Fill an object with a gradient; (1 pt)
  - Delete objects using mouse clicking; (2 pt)
  - Use mouse as an eraser of:
    - a fixed width; (1 pt)
    - a adjustable width; (2 pt)
  - Zoom in and out application working area using keyboard; (2 pt)

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 on 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 aplied 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 specifiv 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 found a way to zoom only the working area.

In the repository i included the .cbp file and the main.cpp file neccessary 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/DEMENCI/WPLabs

3.3 Prove your work with screens

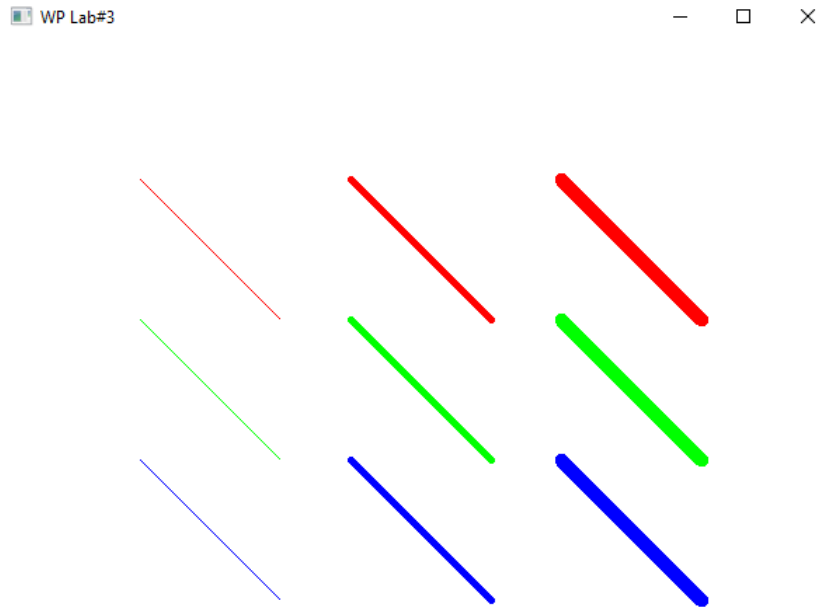

Figure 3.1– Lines of different colors and widths

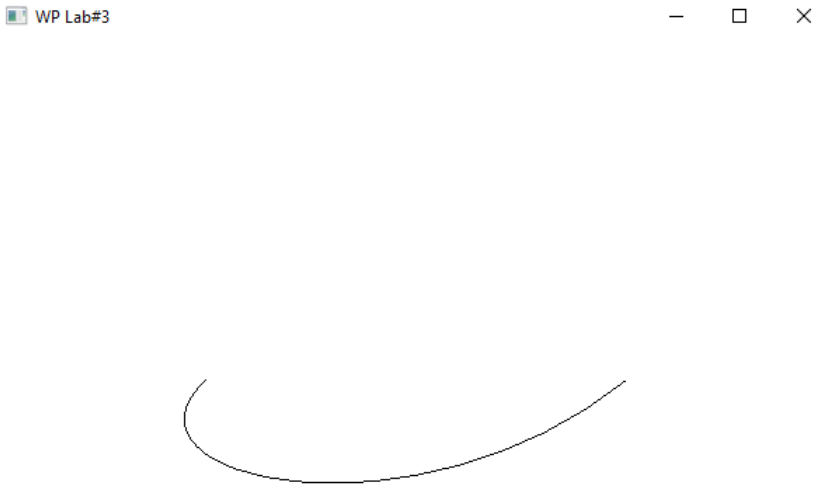

Figure 3.2– Bezier spline

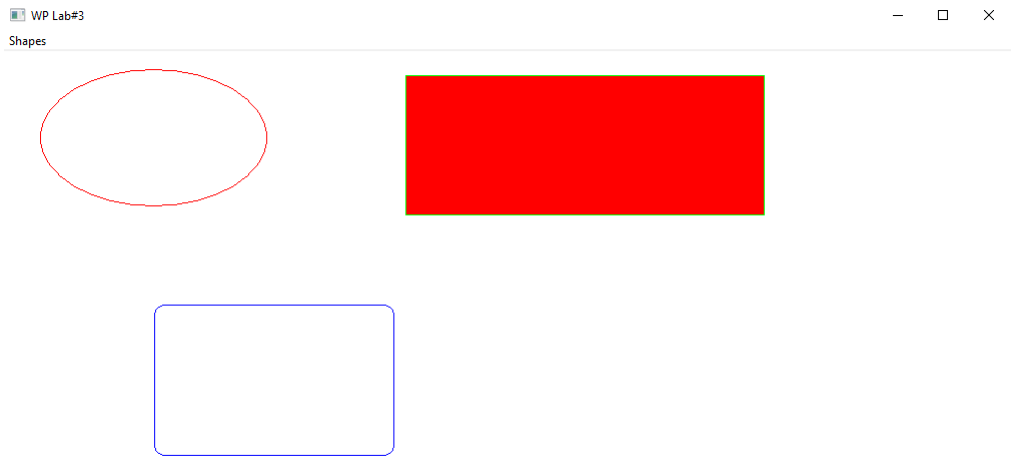

Figure 3.3– Plane objects

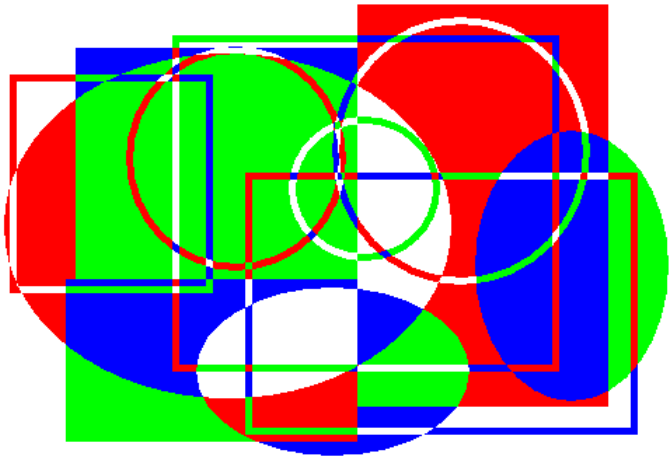

Figure 3.4– Example image

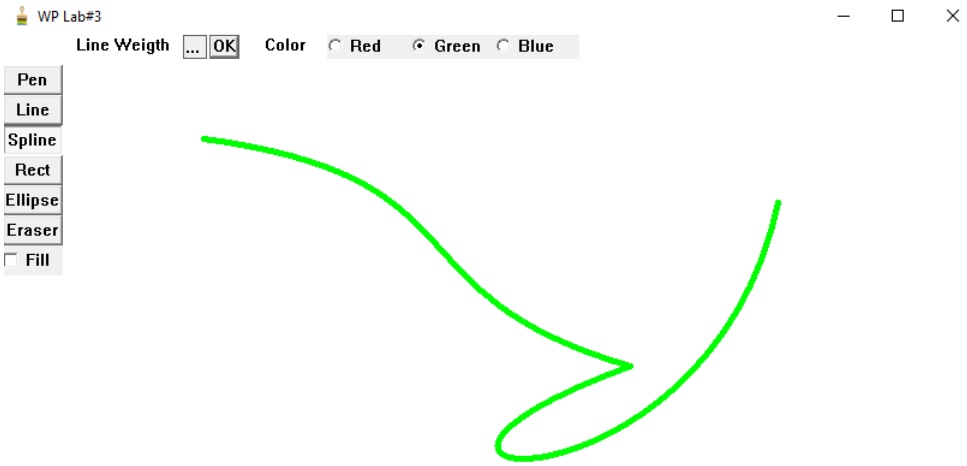

Figure 3.5– Overview of the app

Conclusion

– 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.

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