Faculty of Computers, Informatics and Microelectronics

Technical University of Moldova

WINDOWS PROGRAMMING

Laboratory work #4

# Advanced Form Elements. Child Windows. Basics of Working With Keyboard

*Author:*
Victor Istratii

*Supervisor:*
Irina Cojanu

**Laboratory work #2**

## 1 Purpose of the laboratory

Gain knowledge about child windows and basics of working with keyboard.

## 2 Laboratory Work Requirements

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

a) Create an animation based on Windows timer which involves at least 5 different drawn objects

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

a) Realize the tasks from Basic Level.

b) Increase and decrease animation speed using mouse wheel/from keyboard

c) Solve flicking problem describe in your readme/report the way you had implemented this

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

a) Realize the tasks from Norma Level without Basic Level

b) Add 2 animated objects which will interact with each other. Balls that have different velocity and moving angles. They should behave based on following rules: -At the beginning you should have 3 balls of different colours of the same size -On interaction with each other, if they are of the same class (circle, square), they should change their color and be multiplied. -On interaction with the right and left wall (the margins of the window), they should be transformed into squares. -On interaction with the top and bottom of the window - the figures should increase their velocity. -Please, take into consideration that the user can increase and decrease animation speed using mouse wheel/from keyboard

– **Bonus point task:**

a) Use a scroll bar to scroll through application working space. Scroll should appear only when necessary (eg. when window width is smaller than 300px)

## 3 Laboratory work implementation

### 3.1 Tasks and Points

Advanced Level: -Add 2 animated objects which will interact with each other. Balls that have different velocity and moving angles. They should behave based on following rules: -At the beginning you should have 3 balls of different colours of the same size -On interaction with each other, if they are of the same class (circle, square), they should change their color and be multiplied. -On interaction with the right and left wall (the margins of the window), they should be transformed into squares. -On interaction with the top and bottom of the window - the figures should increase their velocity. -Please, take into consideration that the user can increase and decrease animation speed using mouse wheel/from keyboard

### 3.2 Laboratory work analysis

Repository:
https://github.com/VictorIstratii151/WP-labs

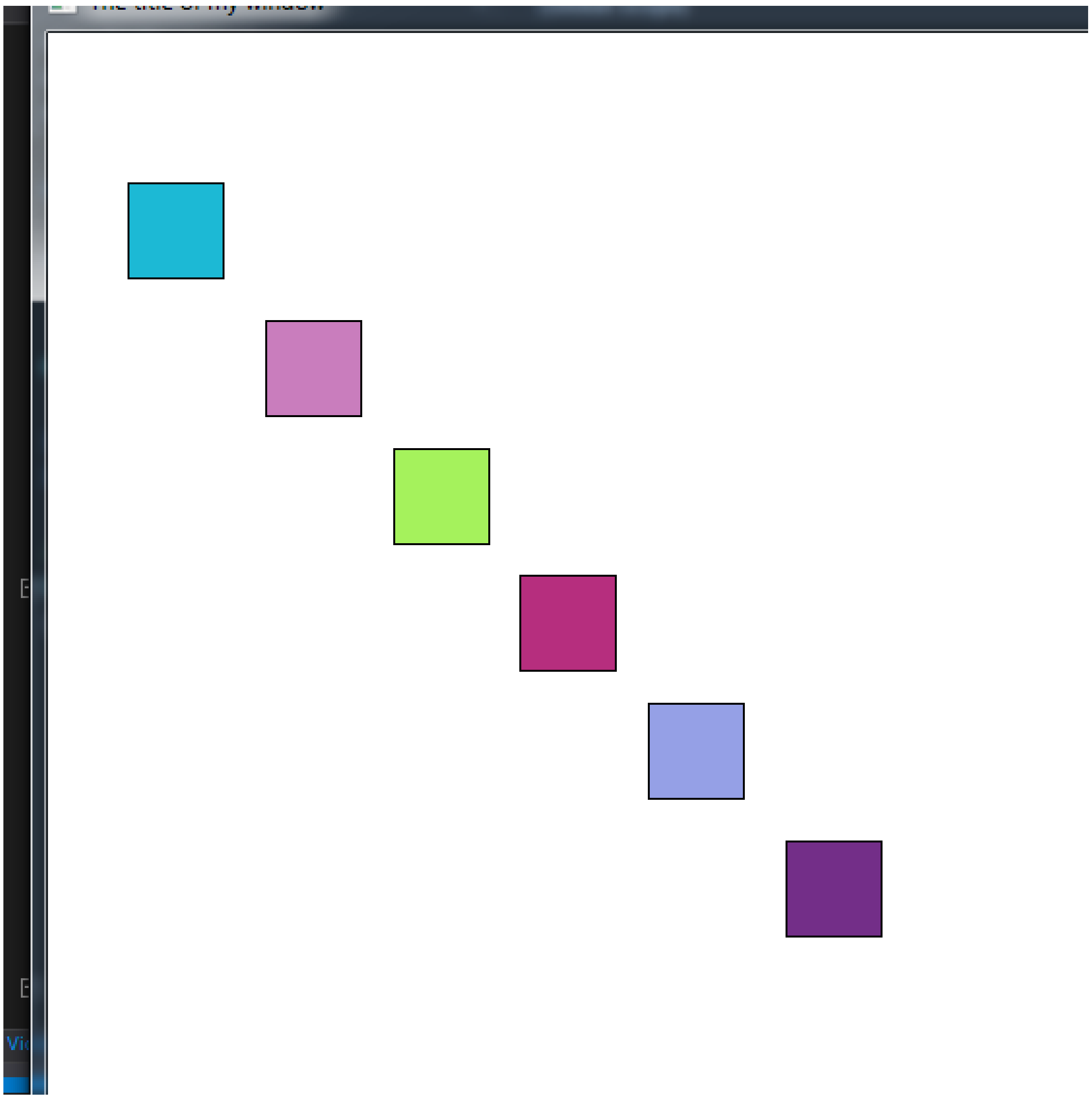
### 3.3 Proving my work

The one thing is annoying about win32 api – when it comes to implement continuous animations of different objects, but more precisely when you draw directly to the HDC of your window, there is a big possibility that the screen will be updated before the drawing is finished. That's why usually there occur strange graphical artefacts, known as flickers. The more drawing operations are performed, the messier looks the screen.

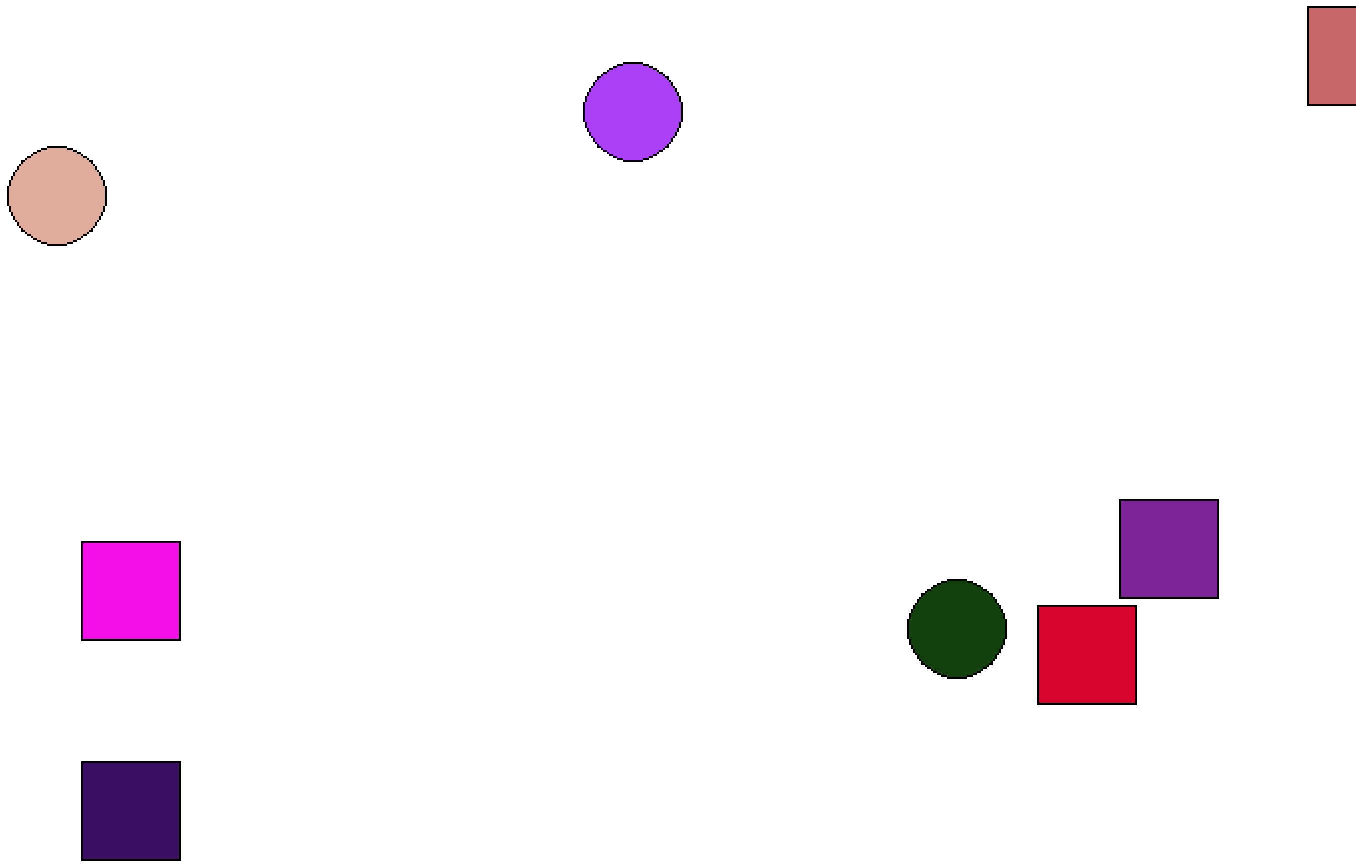
The solution to this problem is pretty easy and is called Double Buffering. This means that we just do all our drawing in memory first, and then copy the completed drawing to the screen, using a single BitBlt() function. That results in the screen getting updated directly from the old image.

To do this, we create a temporary HBITMAP in memory that is the exact size of the area we are going to draw to the screen. We also need a HDC so that we can BitBlt() to the bitmap.

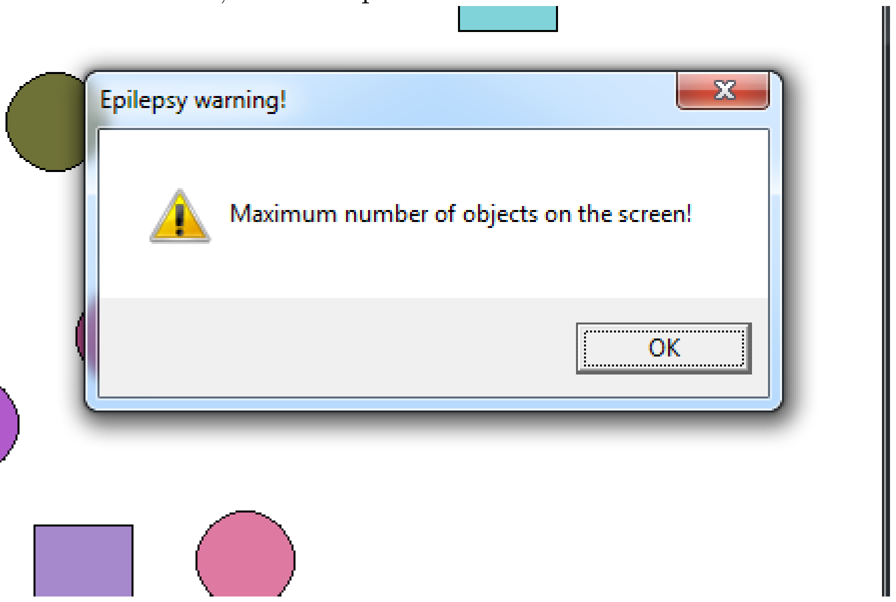
Then, having a place to draw to in memory, all of the drawing operations use a buffer instead of window's hdc, meaning that the results are stored in memory until they are complete.



At first we see that on the screen are created only squares.



Observe that, after some interactions, several squares are transformed to balls.



Epilepsy warning!

Maximum number of objects on the screen!

When objects on the screen are too much, we can't add any more, so it is issued an error.

## Conclusions

While doing these laboratory I have learned about what are the bitmaps, how they are drawn to the screen, and which problems this might cause. Also I have implemented a technique of passing these problems. More than that, from literature I have learned about how we can create a bitmap, which will have the desired region transparent, after applying a mask on it.

I must mention also, that the collision system is not a perfect one. Sometimes it might happen that two or more objects are stuck one in another. This is due to several reasons. For example, when an object is transformed at the exact moment it occurs collision with another object, it may happen that the new shape will overlap with the impacted shape. Also, if we press too fast on the button for adding new balls, with 100 percent probability will occur the overlapping of multiple objects, causing interesting glitches.