|  |  |
| --- | --- |
|  | /storage/emulated/0/.polarisOffice5/polarisTemp/image1.png |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждениевысшего профессионального образования «Московский технологический университет»  МИРЭА | | |

Институт Информационных Технологий

Кафедра Корпоративных Информационных Систем

**ОТЧЕТ**

по Лабораторной Работе №2

на тему

«Основные компоненты библиотеки Windows.Forms»

по дисциплине

«Процедурное Программирование»

Выполнил студент группы ИКБО-07-17 Акжигитов Р.Р.

Принял старший преподаватель Мирзоян Д.И.

Выполнено «16» октября 2017 г.

Зачтено «\_\_\_» \_\_\_\_\_\_\_\_\_ 2017 г.

Москва, 2017

**Задание**

**Часть 1.**

Создать приложение, отображающее движущийся вдоль края окна текст. Текст должен двигаться плавно, не выходя за границы отображаемой области окна (за рамку), не отрываясь от края окна при изменении его размеров. Кроме того, текст должен плавно менять цвет в соответствии с вариантом. Двигаться текст должен по часовой стрелке (От левого верхнего угла к правому верхнему, от правого верхнего к правому нижнему, от правого нижнего к левому нижнему, от левого нижнего к левому верхнему).

**Часть 2**

Добавить в приложение еще одну форму, которая открывается вместе с первой формой одновременно, поместить на форму компоненты и организовать их взаимодействие в соответствии с индивидуальным заданием.

**Теоретическая часть**

Для создания приложений в Windows.Forms есть удобный конструктор, который включает в себя множество различных по функциональности элементов, таких как кнопки, метки, текстовые поля, таймеры и т.д.

Для создания приложения следует в графическом редакторе поместить нужные элементы, расположив их в программе по желанию или по техническому заданию (также это можно сделать с помощью свойств Left, Top, которые отвечают за X и Y верхнего левого угла компонента).

Каждый из компонентов имеет свои свойства, некоторые уникальные и общие характерные всем компонентам, например, **Name, Location, Size** (характерные для всех отображаемых компонентов).

Частоиспользуемое свойство для **label, textBox** – **Text** (получение текстового содержимого из «текстовых» компонент). В лабораторной работе так же часто использовалось такое свойство, как **Left, Top** для изменения местоположения элемента в форме.

Для **checkBox** часто применяется свойство **checked**, для того, чтобы узнать состояние элемента (Истина – если «активирована», ложь – если «деактивирована»).

Для **listBox** часто используется свойство **Items** для доступа ко всем элементам, содержащимся в данном элементе. А уже к Items (так как это обычная коллекция элементов) мы можем выполнять некоторые полезные методы, такие как Add (для добавления нового элемента в наш компонент), Remove (для удаления элемента, если он был найден, параметром стоит передать строковое представление элемента). Так как в данном компоненте можно выделить некоторые элементы, то для получения выбранного элемента следует использовать **listBox.SelectedItem**.

Важнейшее понятие для компонент – события. На любой компонент можно назначить определенное событие, которое, когда случится (и если случится), будет выполняться определенный код, написанный нами. Существоует множество событий общих для многих компонент, таких как событие клика мышки, двойного нажатия, введение мышь в поле элемента (**Click, DoubleClick, MouseMove, MouseHover, MouseEnter** etc)

Самое частоиспользуемое **button\_Click** для отлавливания события нажатия (мышкой или клавиатурой) на кнопку, в теле события нужно написать нужные операции и операторы.

В данном задании использовалось событие **timer\_Tick**, для срабатывания кода через определенный интервал времени (в частности для изменения координат метки и для изменения ее цвета).

Также использовалось **checkBox\_CheckedChanged** для события активации или деактивации элемента состояния.

И **listBox\_SelectedIndexChanged** для того, чтобы узнать, когда мы выделили элемент в **listBox**.

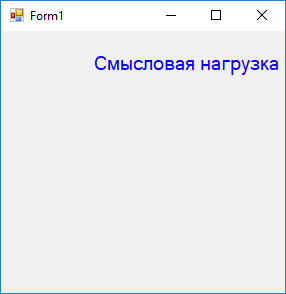
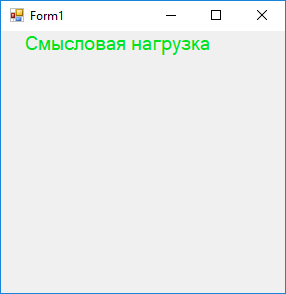
Также использовано событие **Resize** для формы, чтобы в нужный момент пересчитывать координаты метки, чтобы метка двигалась, неотрываясь от границ экрана.

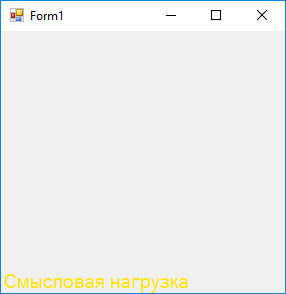
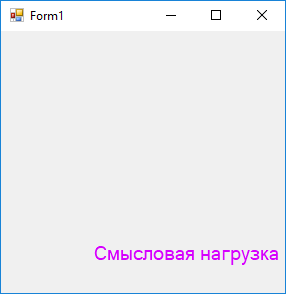
Существует понятие форма, это такие окна (рабочие области), в которых мы размещаем элементы. Их может быть несколько или только одна. В данном задании мы создали 2 формы и связали их с помощью некоторого кода, предоставив друг другу свои свойства (возможность доступа). Очень важно использовать нужный модификатор доступа, например, в элементе таймер, ибо в другой форме он поумолчанию недоступен, нужно изменить доступ на **public**.

**Алгоритм решения задачи**

1. Добавить компоненты на первую форму.
2. Запрограммировать движение label по краям формы (с помощью перечисленния направления движения enum) и события timer.Tick.
3. В зависимости от условий (координат тектовой метки) менять направление движения.
4. С помощью label.ForeColor применить цвет к метке, с помощью и координаты движения, как изменяющийся во времени параметр, запрограммировать изменение цвета, как трех составляющих Color.FromArgb()
5. Связать 2 формы между собой.
6. С помощью обращения из второй формы к первой сделать выключечние таймера, чтобы событие перестало срабатывать.
7. Добавить компоненты для работы с элементами списка.

**Тестирование**





**Заключение**

Получив определенные знания о компонентах Windows.Forms, я узнал о свойствах, событиях, формах, также смог сделать довольно непростое тех. Задание.

**Исходный код**

Код формы 1:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace laba2

{

public partial class Form1 : Form

{

enum Direction { Right, Left, Up, Down };

Direction direction = Direction.Right;

public Form1()

{

InitializeComponent();

}

private void timer1\_Tick(object sender, EventArgs e)

{

if (label1.Top < 0)

{

label1.Top = 0;

direction = Direction.Right;

}

if (label1.Left > ClientRectangle.Width - label1.Width)

{

label1.Left = ClientRectangle.Width - label1.Width;

direction = Direction.Down;

}

if (label1.Top > ClientRectangle.Height - label1.Height)

{

label1.Top = ClientRectangle.Height - label1.Height;

direction = Direction.Left;

}

if (label1.Left < 0)

{

label1.Left = 0;

direction = Direction.Up;

}

switch (direction)

{

case Direction.Right:

label1.ForeColor = Color.FromArgb(0, ((ClientRectangle.Width - label1.Width) - label1.Left) \* 255 / (ClientRectangle.Width - label1.Width), (label1.Left) \* 255 / (ClientRectangle.Width - label1.Width));

label1.Left += 10;

break;

case Direction.Down:

label1.ForeColor = Color.FromArgb(label1.Top \* 255 / (ClientRectangle.Height - label1.Height), 0, 255);

label1.Top += 10;

break;

case Direction.Left:

label1.ForeColor = Color.FromArgb(255, ((ClientRectangle.Width - label1.Width) - label1.Left) \* 255 / (ClientRectangle.Width - label1.Width), label1.Left \* 255 / (ClientRectangle.Width - label1.Width));

label1.Left -= 10;

break;

case Direction.Up:

label1.ForeColor = Color.FromArgb(label1.Top \* 255 / (ClientRectangle.Height - label1.Height), 255, 0);

label1.Top -= 10;

break;

}

}

private void Form1\_Shown(object sender, EventArgs e)

{

Form2 newForm = new Form2();

newForm.Show();

newForm.form1 = this;

}

private void Form1\_Resize(object sender, EventArgs e)

{

switch (direction)

{

case Direction.Right:

label1.Top = 0;

break;

case Direction.Down:

label1.Left = ClientRectangle.Width - label1.Width;

break;

case Direction.Left:

label1.Top = ClientRectangle.Height - label1.Height;

break;

case Direction.Up:

label1.Left = 0;

break;

}

}

}

}

Код формы 2:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace laba2

{

public partial class Form2 : Form

{

public Form1 form1;

public Form2()

{

InitializeComponent();

}

private void checkBox1\_CheckedChanged(object sender, EventArgs e)

{

form1.timer1.Enabled = checkBox1.Checked;

}

private void button1\_Click(object sender, EventArgs e)

{

listBox1.Items.Add(textBox1.Text);

textBox1.Text = "";

}

private void button2\_Click(object sender, EventArgs e)

{

if (listBox1.SelectedIndex < 0)

MessageBox.Show("Вы не выбрали ни один элемент из списка.");

else

listBox1.Items.Remove(listBox1.SelectedItem);

}

private void listBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

if (listBox1.SelectedIndex >= 0)

label1.Text = "Выбранный элемент: " + (listBox1.SelectedIndex + 1) + "; Текст: " + listBox1.SelectedItem.ToString();

else

label1.Text = "Выбранный элемент: Нет";

}

}

}