UDC 004.021

GRAPHIC INTERFACE DESIGNING FOR AUTOMATED INFORMATION ACCOUNTING SYSTEM FOR GOODS TO PROMOTE SMALL BUSINESS OF THE "GRANDDECOR" COMPANY UNDER THE OPERATING SYSTEM OF WINDOWS

MAKSIM TRASHCHANKA, IRYNA BURACHONAK
Polotsk State University, Belarus

This article discusses the principles of building a graphical user interface in the AIS accounting for goods to promote small business of the company "GrandDecor" under the Windows operating system.

Different technologies are used to create graphical interfaces using the .NET platform – Window Forms, WPF, applications for the Windows Store (for Windows 8 / 8.1 / 10). However, the most simple and convenient platform is still Window Forms or forms. This article focuses on the description of technologies for creating graphical interfaces using Windows Forms technology.

A Windows Forms application is an event-driven application supported by the Microsoft .NET Framework. Unlike batch programs, most of the time is spent waiting for the user to take any action, such as typing text in a text field or clicking a button.

A form is a visible surface on which information is displayed to the user. Typically, a Windows Forms application is built by placing controls on a form and writing code to respond to user actions, such as mouse clicks or keystrokes. A control is a separate user interface element for displaying or entering data.

When a user performs an action with a form or one of its controls, an event is created. The application responds to these events with a code and handles events as they occur.

The big drawback of Windows Forms was that the programming and design teams had to work very closely to make a great project. That is, the designer drew the interface, gave it to the programmer, and the programmer, in turn, implemented it (aside from his immediate task - he had to adjust the buttons to fit the dimensions, insert pictures, etc.) rather than implement the logic of the program.

To get rid of this lack of Windows Forms, the Windows Presentation Foundation technology WPF was included in the .NET Framework, a big step towards improving interface design. Compared to Windows Forms - the following was done (roughly speaking) - the programmer was completely immersed in the development of the program logic, and the designer could immediately create the program design, almost independent of the programmer.

The popularity of Windows Forms is fading. But it continues to be used in simple, not requiring great interfaces, programs. Also in all versions of the .NET Framework there is support for Windows Forms, and some additions and improvements are included there.

Despite the fact that in Windows Forms, the entire interface can only be built using the mouse, the program code markup is also available for the programmer, and can be changed if desired.

Sample code for a WF designer is shown in listing 1.

Listing 1 – Sample WF Designer Code

```
namespace HelloApp
{
  partial class Form1
  {
    private System.ComponentModel.IContainer components = null;
    protected override void Dispose(bool disposing)
    {
      if (disposing && (components != null))
      {
        components.Dispose();
      }
      base.Dispose(disposing);
    }
    #region Windows Form Designer generated code
    private void InitializeComponent()
```

```
    {
        this.SuspendLayout();
        this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
        this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
        this.ClientSize = new System.Drawing.Size(284, 261);
        this.Name = "Form1";
        this.Text = "Hello world!";
        this.ResumeLayout(false);
    }
    #endregion
  }
}
```

Using the special Properties window on the right, Visual Studio provides us with a convenient interface for managing the properties of an element:

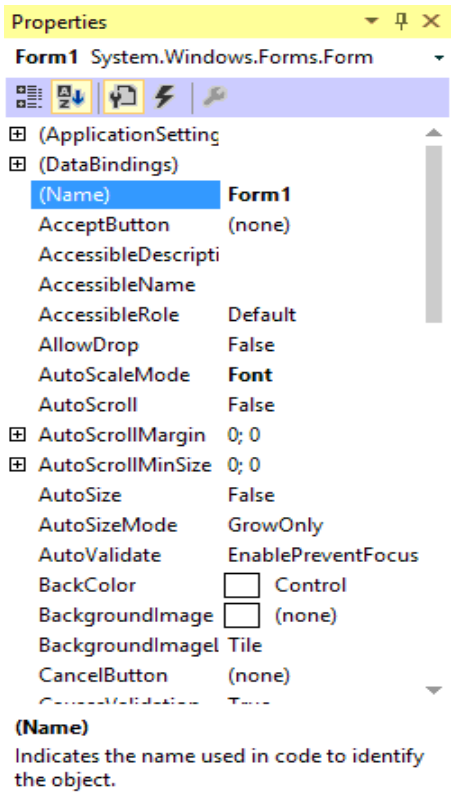

Figure 1. – Properties window

Most of these properties affect the visual display of the form. We analyze the basic properties:

– Name: sets the name of the form - or rather the name of the class that inherits from the Form class

– BackColor: indicates the background color of the form. By clicking on this property, we can choose the color that suits us from the list of suggested colors or the color palette.

– BackgroundImage: indicates a background image of a form

– BackgroundImageLayout: determines how the image specified in the BackgroundImage property will be placed on the form.

– ControlBox: indicates whether the form menu is displayed. In this case, the menu refers to the top-level menu, where the application icon, the form header, and the form minimization buttons and the cross are located. If this property is false, then we will not see either the icon or the cross, which usually closes the form

– Cursor: determines the type of cursor that is used on the form

– Enabled: if this property is false, then it will not be able to receive input from the user, that is, we will not be able to click on the buttons, enter text in text fields, etc.

– Font: sets the font for the entire form and all controls placed on it. However, by setting the font of the form elements, we can thereby override it.
– ForeColor: font color on the form
– FormBorderStyle: specifies how the form border and title bar will be displayed. By setting this property to None, you can create the appearance of a free-form application.
– HelpButton: indicates whether the form's help button is displayed.
– Icon: sets the shape icon
– Location: determines the position relative to the upper left corner of the screen when the StartPosition property is set to Manual
– MaximizeBox: indicates whether the maximize window button will be available in the form header
– MinimizeBox: indicates whether the minimize button will be available
– MaximumSize: sets the maximum size of the form
– MinimumSize: sets the minimum size of the form
– Opacity: sets form transparency
– Size: defines the initial size of the form
– StartPosition: indicates the initial position from which the form appears on the screen
– Text: defines the form header
– TopMost: if this property is true, then the form will always be on top of other windows
– Visible: whether the form is visible, if we want to hide the form from the user, then we can set this property to false
– WindowState: indicates the state in which the form will be on startup: normal, maximized or minimized

This article discusses the positive aspects of using Windows Forms, the possibilities for a fast and flexible interface development, and discusses the basic properties of graphic elements.

REFERENCES

1. C# Windows Forms Application Tutorial with Example. [Electronic resource]. – Access mode: https://www.guru99.com/c-sharp-windows-forms-application.html/.– Access date: 18.11.2018.
2. Пошаговые руководства по Windows Forms. [Electronic resource] / CitForum. – Access mode: https://msdn.microsoft.com/ru-ru/library/zftbwa2b(v=vs.110).aspx/. – Access date: 19.11.2018.
3. Руководство по программированию в Windows Forms.[Electronic resource] / Metanit. – Access mode: https://metanit.com/sharp/windowsforms/. – Access date: 21.11.2018