Mykhaylo KOZELKO
UKRAINE ACADEMY OF PRINTING,
Pid Goloskom Str. 19, 79020 LVIV

Designing of graphic user interface based on abstract elements

M.Sc. eng. Mykhaylo KOZELKO

He specializes in theoretical and applied computer science, theory of algorithms, programming, information systems, mathematical modeling of systems, computer simulation and mathematical modeling.

e-mail: actionmike@mail.ru

Abstract

The paper deals with information technology building graphical user interfaces (GUI) of Computer Systems using an abstract graphic element. There are introduced and applied terms "abstract graphical interface" and "abstract graphic element". Principles of their construction and operation algorithms are also shown. The abstract interface is general, and may be adapted for specific cases. Design of any specific interface by means of the use the abstract interface is easier, because general form is common, and only specific details must be developed. In this exists the advantage of abstract graphic element over other existing means of interface forming.

Keywords: GUI, graphic elements, graphic element model, interface graphic editor, informatics tools.

Projektowanie graficznych interfejsów użytkownika przy użyciu elementów abstrakcyjnych

Streszczenie

Artykuł dotyczy technologii informatycznej komputerowych systemów interfejsów graficznych przy użyciu graficznego elementu abstrakcyjnego. Wprowadzono i wyjaśniono terminy "abstrakcyjny element graficzny" i "abstrakcyjny interfejs graficzny". Utworzono model matematyczny technologii informatycznej do formowania elementu abstrakcyjnego i abstrakcyjnego interfejsu użytkownika oraz model komputerowego systemu tworzenia elementów abstrakcyjnych i abstrakcyjnych interfejsów użytkownika dla systemów informatycznych. Model daje możliwość automatycznego przekształcenia elementów abstrakcyjnych do typowych graficznych elementów interfejsów użytkownika i tym samym stworzenia różnorodnych interfejsów na podstawie jednego interfejsu abstrakcyjnego. Zostały wskazane zalety stworzonej technologii informatycznej w porównaniu do technologii istniejących. Do syntezy modelu matematycznego zostały zastosowane metody dekompozycji i zmodyfikowanej algebry algorytmów. Model ogólny i modele poszczególnych podsystemów podane zostały z wykorzystaniem operacji sekwencjonowania zmodyfikowanej przez wprowadzenie dodatkowych właściwości, określonych w zmodyfikowanej algebrze algorytmów. Na podstawie modelu został zbudowany komputerowy system tworzenia elementów abstrakcyjnych i abstrakcyjnych interfejsów użytkownika systemów informatycznych. Model zaimplementowano na platformie Microsoft Visual Studio .NET z wykorzystaniem języka programowania C# i języka markowania XAML.

Słowa kluczowe: graficzny interfejs użytkownika, GUI, element graficzny, model elementu graficznego, edytor interfejsu graficznego, narzędzia informatyczne.

1. Introduction

Nowadays, the functional features of Internet websites or web projects are similar to the desktop systems. From the other side, modern systems of developing and design of graphical interfaces allow create projects and use graphic elements only of strict types. So, there is evident a need for the Editor that would help the developing a graphic user interface (GUI) for many different types of projects, and input into the design the element, that would not limit the user's choice of graphic elements and allow designing new ones. So the article focuses on resolving issue of modern editor in GUIs based on abstract interface and elements.

2. Terms of abstract graphical interface and an abstract graphic element

Abstract GUI - is a graphical user interface in the design phase which is does not specified what type of computer technology (desktop or web) it will used. The types of graphic elements are determined after the completion of the interface design. This approach makes it possible to use the same graphical interface to the desktop and web applications, and later on possibly in mobile applications.

Abstract graphic element – is a graphics primitive constructed as a rectangle that hosts the text label to display its name and that can be converted to any known type of graphic element (button, text box, label, panel ...). Abstract graphic element is characterized by properties (height, length, color, transparency, visibility, coordinates location...), and events that are executed over the abstract element (click, double click, download...), and by methods of their processing. Abstract graphic element is part of Module that can contain other graphic elements including abstracts . The use of abstract graphic elements in the development of graphical interfaces of Computer Systems has several advantages when replacing a specific graphic element:

- there is no need to create a new graphic element, it is sufficient to convert an abstract element to the selected type (panel, grid, …);
- there is no need to re-define properties and events of the element, they are automatically saved during converting the abstract into specific element;
- there is no need to re-set methods for handling events for the new element;
- there are no errors in the code when replacing the abstract element into specified type; the code is changed automatically;
- after conversion, all subsidiary elements that are placed in an abstract graphic element become subsidiaries of the selected one.

Both abstract graphic element and abstract graphical interface can be used for designing prototypes of future interfaces at minimal cost for their implementation, because this prototype is sufficient to convert to the selected type, and its elements to the needed elements.

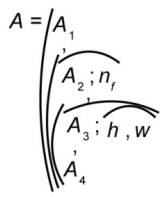
3. The model of information technology

The model of informatics technology for graphic element building by the algebra of algorithms [1, 2] is described by the formula (1).

$$S = \overset{\frown}{A\,;\,B\,;\,C\,;\,D\,;\,E\,;\,F} \tag{1}$$

where: A – subsystem of an element creation of interface form, B – subsystem of generation the graphic elements of interface, C – subsystem of processing operations over the graphic elements and providing them with functional characteristics, D – subsystem of converting abstract elements to typical elements of GUI, E – subsystem of generation the created project to Windows or Web Project, F – subsystem of compilation by the created project.

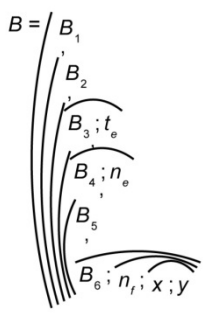
Subsystem A creates the new element of form. The mathematical model of the creation subsystem of the form element is described by formula (2) of algebra of algorithms.



(2)

The subsystem contains a graphic element A_1, the choice of which is initializing by a new form element. According to the fact that user can create many form elements thereby realizing a multi-window project, each created form element is specified by the element name n_f. The name of form element n_f is necessary for identification the elements of the form when placing on it the graphic elements, and forming a hierarchical tree of displaying the form elements, namely the sequence of call and display form elements in multiple-windows project. To define the name n_f the form element A_2 is intended. After the user selects a graphical element A_1 the form element appears on the display. The next element of the subsystem creating the form element is a functional element A_3, which describes a direct creation of the form element. Functional element A_3 is characterized by properties h and w, needed to define the size of the element shape, height and length, respectively. After creation the form element it is located and displayed in graphics editor. Display a form element is realized by functional element of A_4.

Subsystem of generation a graphical element consists of several functional elements and is described by tools of algebra algorithms by the formula (3).



(3)

The structure of subsystem B contains the list of graphical elements B_1 that consists of all available graphic elements of a particular type and it also consists of an abstract element. The list of graphical elements B_1 is designed to select a graphic element of particular type, its further generation and placement on menu form. The functional element B_2 is used to generate an abstract graphic element. B_2 element creates an instance of an abstract element, the initial values of the properties of element (height, width, color etc). Generating the graphical element of known type is realized by functional element B_3. The principle of constructing a graphical element is the same as the construction of an abstract element. However, there is one difference - functional element B_2 generates only an abstract element, and functional element B_3 gives the type of graphic element t_e, and according to the value of t_e is created an instance of the graphic element of specified type. The value of a type element t_e is determined by the choice by the user from the list of graphic elements B_2, the element of the desired type. To implement the possibility to change properties of a particular graphical element and to set procedures for handling events, every graphic element must have a unique name to identify it. The name of abstract or an abstract graphic element is specified in B_4. The name is displayed on the display when selecting an item from the list of B_1. The functional part of the B_5 sets a generated graphic element the function according to the selection by user to define or edit the properties of the element. The place of the created graphic element is provided by a functional element of B_6. B_6 component contains two parameters: element name form n_f and coordinates of the location x and y of the graphic element on the menu form. n_f is the name of the active form element on which are created graphic elements. According to the fact that user can generate several forms, this implementation of subsystem, and in particular the functional element B_6, provides proper placement of graphic elements created in the right form.

After the user selects a graphical element, that is located on the element form, it is further processed by the subsystem C. Functional element C_1 determines the proper name of graphical element n_e in order to apply changes of graphical element's properties to the selected graphical element. Functional element C_2 determines the type of the selected graphical element t_e. Determining the type t_e is necessary to define all properties that are inherent to the selected graphical element. A list of all available properties of the C_{2n} is formed based on the type of selected graphical element, and the list C_{2v} from values of the relevant properties from the list C_{2n}. The values of list C_{2v} are available for editing. The use of new values to properties of selected graphical element occurs automatically when you changes the values of properties in the list C_{2v}. Setting the functional features of the graphical element is similar to editing the properties of the graphical element. Functional element C_3 determines the type t_e of selected graphical element. A list C_{3n} of all available actions on the graphical element of the selected type t_e is formed, and a list C_{3v} of fields of inputting the name of processing methods of relevant events from the list C_{3n}. is also formed. After entering the name of the method in the field of the selected event, this method is automatically assigned to the chosen event of graphical element. Thus, the graphical element sets the functional features - upon the occurrence of certain events is an appeal to the appropriate method, where certain information is processed. Some properties of a graphical element (such as height, width, coordinates of the location on the element form) subsystem C allow to edit, to make changes of the size and position of graphical element with the mouse, and to apply these actions directly to the selected graphical element. This method probably executes changes of sizes and position of graphical element, because there is no need to apply to the list of the element's properties and look for the needed one in a dozen of all available properties. This method is implemented by functional element C_4. Element C_4 is characterized by properties of height h of a graphical element, width w, distance at which is located a graphical element on form element or other graphical element from the left edge x and top retreat y.

User can convert the abstract elements into typical graphical elements. Subsystem D_1 is designed to convert abstract graphical elements. After selection the graphical element with the help of functional element D, the determination of type of selected graphical element happens. Then D subsystem checks the conditions of ownership type of selected graphic element to the abstract type t_e=*abstract*, conversion the selected element is possible only under condition. All graphical elements can be divided into two groups - graphical elements which can be placed on other graphical elements (container graphical elements), and graphical elements on which it is impossible to locate other elements (subsidiary graphical elements). Given this feature of graphical elements and that user is able to place on an abstract graphical element other graphical elements, functional element D_2 determines the number of subsidiaries graphical elements k. Depending on the value of subsidiary graphical elements that are located on the selected abstract graphical element, the formation of available types of graphical elements to which an abstract element can be converted happens. If $k = 0$ then a list of all

possible types of graphical elements D_3 for conversion an abstract graphical element is formed. When on an abstract graphical element are placed other subsidiary elements with respect to an abstract element, namely $k>0$, a list D_3 only with the elements of the container type is formed. If all the above described conditions are met, then functional element D_5 converts an abstract element into typical graphical element. Mathematical description of subsystem D by tools of algebra of algorithms is described by the formula (4).

$$D = \left(D_1 ; t_e \right) \left(\not\in t_e = abstract \right) \left(D_2 ; k \right) \left(\not\in k=0 \; D_3 \right) \left(\not\in k>0 \; D_4 \right) D_5 \tag{4}$$

4. The model of instrumental tools of implementing the information technology

Developed model has a graphical user interface with a set of tools for processing and displaying the information that are designed for maximum user convenience. In graphic systems, the user interface is implemented by single window or multiple windows mode, change the color, size, visibility (transparency, translucency, invisibility) of windows and their elements, their location, sorting elements of windows, flexible configuration of both the windows and some of their elements (files, folders, shortcuts, fonts, etc.).

Developed model consists of six major subsystems that create instrumental tools of development the GUI of computer programs, specifying their functional features. The scheme of instrumental tools of synthesis the elements GUI is shown in Figure 1. This scheme consists of blocks which are subsystems of the editor graphical elements, such as:

PUF – subsystem elements of the form;

PGU – subsystem of generation graphic and abstract graphic elements;

PV – subsystem of displaying and changing the properties of graphic elements;

PP – subsystem of displaying and setting methods for handling events graphic elements;

PC – subsystem of converting abstract elements to typical graphic elements;

PZ – subsystem of storage and compilation the created project.

The diagram also shows the relationships between its subsystems and editor of text elements (RTU).

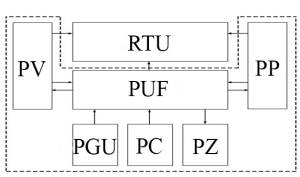


Fig. 1. The scheme of instrumental tools of GUI element construction
Rys. 1. Schemat narzędzia do projektowania graficznego interfejsu użytkownika

Subsystem elements of the form (PUF) are designed to generate form elements and then placing on them the specified graphic and abstract elements. Subsystem PGU is used to generate graphic and abstract graphic elements and then placing them on the elements of the form. PGU subsystem consists of a toolbar on which are placed controls to create common graphical objects such as buttons, text box, label, etc., and abstract graphic elements. It's possible to create a graphic or an abstract element by two ways - pressing the button on the controls or by dragging it with your mouse on the base field. Displaying and changing the properties of typical graphic and abstract elements is implemented by PV subsystem. It is formed by a list of fields that are intended to display the names of the properties and controls, that show the values of the properties with the possibility of changing these values and automatically assigning new values to the relevant properties of graphic elements. PP subsystem is designed to reflect the events of abstract or graphic elements and name list of these events. It consists of a list of all available events that are inherent to the selected graphic elements and fields designated to specify and display the defined procedures for processing the relevant events of graphic elements. Converting an abstract graphic element to a typical one is made up by subsystem of the PC. Conversion is carried out after selecting an abstract graphic element. PC subsystem is formed by the list of controls, each of which is designed to convert an abstract element to a certain type of graphic element, and by the element of conversion confirmation of an abstract element. The list of controls is formed depending on the presence of subsidiary elements on abstract elements. If an abstract element contains subsidiary elements, so then is formed a list of controls that generates only such graphic elements that can contain other subsidiary elements, e.g. button, panel, grid, stack panel. If the abstract element contains no subsidiary graphic elements, so then is formed a list that consists of all items that are available for conversion. PZ subsystem is used to store the finished project, its compilation and run to run. It's possible to save the developed project as a desktop computer system or as an internet system.

5. Conclusions

1. Introduction of an abstract graphic element provides the construction of abstract graphical interfaces with the ability of their conversion in a number of graphical interfaces. The use of abstract elements during the development of the interface provides an opportunity to abandon the strictly defined graphic elements.

2. Created and implemented model of instrumental tools provides greater opportunities in the development of interfaces and reduces the time of their creation.

3. Developed and implemented model of information technology of creation the abstract graphical interfaces of computer systems is versatile, flexible with the ability to re-use, designed interfaces for different types of projects.

Created models of information technology and instrumental tools are implemented on the platform Microsoft Visual Studio. NET using the language of object-oriented programming C# and markup language XAML [3, 4].

6. References

[1] Ovsyak A., Ovsyak V.: The extended algebra of algorithms with additional cycle elimination axiomats. Conf. "Intelligent Information and Engineering Systems" (INFOS 2011), Sept. 19-23, 2011, Polańczyk, Poland, pp. 23 – 34.

[2] Ovsyak A., Ovsyak V.: The extended algebra of algorithms with multiconditional elimination. Bulletin of the National University "Lviv Polytechnic": Computer Science and Information Technology, No 672, 2010, pp. 291 - 300.

[3] Shield G.: C # 4.0: Full, leadership. Moscow: OOO Y.D. "Williams", 2011, pp. 1056.

[4] McDonald M.: WPF: Windows Presentation Foundation. In. NET 3.5 with Example in C # 2008 for professionals. Moscow: OOO Y.D. "Williams", 2008, pp. 928.

otrzymano / received: 29.08.2013
przyjęto do druku / accepted: 01.10.2013 *artykuł recenzowany / revised paper*