V. V. Voytko, Sc. (Eng.), Assist. Prof.; A. V. Denisyuk; G. L. Lutsyshin

UNIVERSAL GRAPHIC COMPONENT EDITOR GAMING CARDS

The paper suggests the developed version of graphical editor game cards, realized by architectural principle of objects management applying component technology to enhance its functionality in open-type systems. The editor is characterized by independence on the subjects and genres of games and game-specific frameworks, hence providing the possibility of its universal use not only in the field of game technologies, but also as a tool for modeling systems of objects in design and engineering projects.

***Keywords**: game framework, automation of games design, game essence, component architecture, universal characteristics, structure of graphic games.*

Introduction

Today, in time of rapid development of computer-based technologies, multimedia products aimed at achieving the research, scientific, educational, promotional, entertainment and commercial purposes, are widely used. The development of modern computer games is preceded by the creation of game cards editor, which requires additional time within forecast period of the end product development [1]. Game cards designed by manufactures are implemented as separate files, formats of which are not standardized and are not disclosed. Editor of game cards may be included in the basic game delivery to provide the possibility of creation and usage of their own cards within a separate multimedia product in order to implement the authors ideas in graphical environment (such opportunity is provided in games Termins, Lode Runner, Serious Sam, etc.).

However, modern editors of game cards are characterized by highly specialized features, limited and strictly defined functional possibilities and orientation on specific software to *create computer applications, which limits future use of existing graphical editors even for* writing new versions of multimedia products. Using existing editors of game cards by other developers in general is considered to be impossible because of the orientation of the editors at different game frames. Therefore, development of graphical editors of game cards, with special characteristics of universal nature, which enables their usage in creation of computer games is very actual problem.

Hence, the aim of research is automation of game cards creation by means of universal graphic editor. Object of study - methods of graphic editors creation. Subject of research are principles of game cards creation and methods of realization of graphic effects in computer games. The main tasks of the research we consider the development of graphic editor of game cards and provision of its universal methods of graphical editors construction

Analysis of modern graphics editors of game cards

Let us consider the possibilities and limitations of some existing editors. Game Termins is – two-dimensional graphic real-time strategy, delivered with built-in card editor. It enables to edit and create cards which will be used in the game. The editor includes a standard set of game objects, oriented at provision of game story. The set is not subject to any modifications and is finite. In addition, there exist certain rules regarding the placement of objects on the card. Thus, the editor is highly specialized in nature, has a hidden format of cards, does not contain the facilities of modifications or addition to the library, all this makes it impossible to be used by other developers in the process of creating new games.

Now let us consider a more powerful editor that is delivered with the game Heroes 3. The editor allows to build-in the game card with pre-defined sets of objects – a set of roads, barriers, trees, grass, etc. The objects in each set have fixed properties and clearly defined designation. Creation of your own objects is not provided. The editor supports only a few fixed sizes of cards. File format of the card is hidden. Therefore, the application of the editor to create your own games is also impossible.

You should pay attention to the fact that today there appeared systems of automated game design. They include Microsoft XNA Game Studio designed to develop games for consoles Xbox and OS Windows. The system has developed levels editor. Unique in its kind is system SkyStudio manufactured by Skyfallen Entertainment, which is a versatile integrated game development environment, independent on the platform and game carcass; generates programming code and has a wide range of possibilities. However, such modern environments has not yet gained popularity and spread, and vast majority of games are written traditionally, using independently developed levels editor. Automated systems have their own scripting languages, techniques of physical phenomena programming, logic and graphics, but have certain limitations on the expansion and improvement of standard objects libraries. Thus, consider the development of the graphic editor game cards, the model which will be open to be actual, it will provide the possibility of expansion of objects set by own COM components and ensure the universality of the editor characteristics.

Problem set-up

The notion of universality of the developed editor implies: enlargement of functional possibilities, independence on genres and themes of games; the ability to adapt for specific situations, independence on the game carcasses. Privacy of input file format of the card is realized in order to protect copyrights, but the editor should be provided with means of reading the card file.

Support of functionality of the editor extension is that the editor is a set of COM components [2], which interact with each other through documented interfaces. Thus, any component can be easily replaced by another, more modernized, or specialized. In addition, the possibility to add new components created by game developers, to configure the editor to fit your needs is provided. The added components are enlisted in the library for reuse. Therefore, the developed editor must have full component architecture.

All modern games have common structure [3]. Game it is a set of objects that interact with each other. Each object has a certain set of properties and logic of behavior. Independence of developed editor on genres and themes of games is in the ability to create new game objects, set them certain properties and logic of behavior and use in the process of game cards composition (even the object "grass" on the card may have characteristics: color, danger, not to mention more complex objects) [3].

Ability to customize the editor for the description of situations with subsequent transformation of universal features in specialized ones in our opinion, is main characteristic of the developed editor. On one hand, the editor provides the possibility to create different game objects, but it is allowed only at the stage of game model development. End version of the editor should be delivered in a specialized form to ensure the correctness of play operation mode. This problem can be solved in two ways: by providing the possibility of components replacement (this approach envisages the creation of two copies of the components: universal - for game development stage and specialized - for the end product), which, in its turn, requires redundant system resources; the creation of components, which via configuration files, can be configured for different modes of operation, and the game developer is free to define the syntax of these configuration files.

Today, there exist many game frameworks, different by principles of operation and use, as well as their specialized capabilities [4]. Graphic effects, supported by a single framework can not be supported by other frameworks, or look differently. If the graphic editor works on certain game framework, the same framework will limit it. If the developer uses different framework of the game, then he will have to create graphic effects on two different frameworks: for the editor and the game itself. It was therefore decided that the editor should not be oriented by default on specific framework. The developer creates the editor component of framework initialization by means of its own framework. Moreover, this component will become common not only for a particular developer, but for all the developers who use this game framework. Thus, it can be stated that the developed editor will be an over structure on a certain game framework. It should be noted that the game frameworks are higher-level game development tools, that is why the creation of the initialization component is not a difficult task.

The format of the input file of the card should be hidden. This will allow the designers to create game cards for existing games without copyright infringement. In the editor, two mandatory system component are provided, one of which keeps the card in the file, and another reads out the information. Of course, like any other component of the editor, these components can be further improved or replaced. Reading component of editor card file will be used directly in the game. Similarly, it is also possible to use any other components of the editor. Thus, the developer of the game only once, must implement necessary graphic and other features of the card by means of its framework, software support for their use is provided by automation facilities of graphics editor.

Development of the editor structure

The basic component of the graphic editor is the register of other components UCWERegistrar. This component contains a dynamic list of components, which at the time of the editor operation are in the active mode. This provides the principle: "every component can be registered in the list and communicate with each active component of the library”. Due to this principle the possibility of the editor can be expanded in any direction. Interfaces of UCWERegistrar component are given in Table. 1.

Table 1

Interfaces of UCWERegistrar component

Name of interface	Name of the function	Functions designation
ICUWERegistrar	Registering	Called by a component for its registration in the list of components
	UnRegistering	Called by a component to remove from the list of registered components
	GetComponent	Registered component always contains a reference to the interface ICUWERegistrar. Through this function it may request the IUnknown interface of any other registered component.
	GetComponentsList	Returns a list of all active components of the editor

Main component of the editor is component UCWEEditor, which creates a window of the main menu of the program and is responsible for loading of other components, listed in the configuration file. One of the interfaces of this component is the interface IUCWEComponent. This is the main interface, which should be implemented by all components, to be supported by the editor. Interface IUCWEComponent contains basic functions of component management.

Another supported interface is IUCWEMenu. Its presence indicates the support by the component of the main menu, and other components can add in the menu own options, using the interface IUCWEMenuItem. Interface IUCEMenu keeps a list of interfaces IUCWEMenuItem of editor components and provides calls of their functions in accordance with the selected menu item. The list of interfaces supported by component UCWEEditor is shown in Table. 2.

Table 2

Name of the interface	Name of the function	Functions designation
IUCWEComponent (function of this component is similar for each component of the editor)	Registering	Any component – loader must call this function of loaded component to load the component can register calling ICUWERegistrar.Registering
	Init	After call of Registering component loader must call this function of loaded component, so the latter could be initialized
	Do	Function is called by components loaders so that components loaded by them components could perform their functions
	UnInit	Deinitialization function. Called by the component loader.
	UnRegistering	Function of deregistration. Called by the component loader so that the loaded component could release previously occupied component UCWERegistrar.
	GetData	Function is called by the component for saving the updated data of the card on the disk
	SetData	The function is called when the loader of the card read the data of the corresponding component and passes the latter card data saved by on the disk by means of the function GetData
IUCWEMenu	AddItem	Any component having the interface pointer IUCWEMenu of other component, can add into menu, its own options, passing in the function AddItem implemented interface IUCWEMenuItem
	DeleteItem	Component, which added an option in the menu of another component, may remove its option, by calling this function

Component UCWEMap adds options of opening and creation of the card in the menu of component UCWEEditor, implements these options and can download other components. UCWEMap at the beginning of its work load three components: UCWEEngine, UCWESaver and UCWEOpener. UCWEEngine – is component of game framework initialization and processing of data entry, the possibilities of which the user will apply in the editor in the process of cards creation. UCWESaver saves the game card in a file on disk, and UCWEOpener is responsible for the correct download. UCWEOpener is used directly in the game to load cards. UCWEMap supports the interface IUCWEComponent, as a component UCWEEditor, and has similar designation of interface functions.

When choosing in the menu component UCWEEditor the options of creation or opening of the card UCWEMap loads components UCWEObjectsManager and UCWELayersManager. UCWEObjectsManager is intended for creation, storage and removal of game objects. The component also saves a list of game objects issues that the user can select and place on the card field. Fig. 1 illustrates possibilities of dialog window of component UCWEObjectsManager.

.

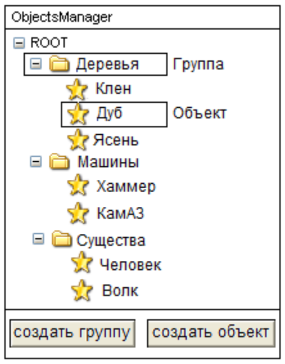

Fig. 1. Dialog window of UCWEObjectsManager component

All the objects in the list are divided into groups. The group does not contain the final information and is not a component, and serves for convenient visual ordering of objects in the list. Interfaces, which support component UCWEObjectsManager, are given in Table. 3.

Table 3

Interfaces of the component UCWEObjectsManager

Name of the interface	Name of the function	Function designation
IUCWEComponent	Interface functions are described in Table. 2	
IUCWEObjectsManager	CopyCurrentObject	Returns a copy of the selected object in the list of object issues
	GetCurrentObject	Returns a reference to the selected object in the list of object issues
	GetObjectsList	Returns a list of object issues

Each object will be initialized in the editor by component UCWEObject, ie UCWEObjectsManager saves the list of components UCWEObject. User in the process of game card creation selects needed object and places it in the card. The structure of the card consists of layers. At certain moment of time there is only one active layer, that is why any added object will belong to one particular layer. Layers provide a means of grouping objects added to the card to arrange a convenient scheme management. Layer in the editor is presented by component UCWELayer. Component UCWELayersManager is responsible for layers editing, which saves the list of all created layers. Dialog window of component UCWELayersManager is shown in Fig. 2.

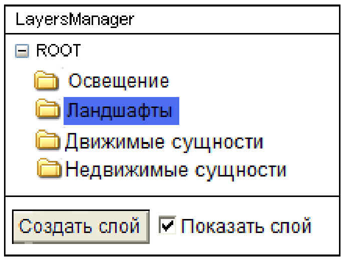

Fig. 2. Dialog component UCWELayersManager

Interfaces, which support component UCWELayersManager, are shown in Table. 4yu

Table 4

Interfaces of UCWELayersManager component

Name of interface	Name of function	Function designation
IUCWEComponent	Interface Options are described in Table. 2	
IUCWELayersManager	GetCurrentLayer	Returns a pointer to the interface of IUCWELayer component, which initializes the current working layer

Component UCWELayersManager manages the list of components of UCWELayer. Component UCWELayer will initialize a specific layer of the card. It contains a list of components UCWEObject - copies of objects from the list in UCWEObjectsManager, added to the card. User action is intercepted on the card by the component UCWEEngine, which addresses to the corresponding function of UCWELayer, and it, in its turn, adds to its list new object which is passed to it as an argument.

When the user presses Delete, the component UCWEEngine calles the required function of component UCWELayer, which removes from the active list the specific (selected) object. Similar actions are taking place all over the card objects. Component UCWEEngine monitors the introduction of user data and calling functions of corresponding components, operates the objects on the card. The list of interfaces of UCWELayer component is shown in Table. 5.

Table 5

The list of interfaces of UCWELayer component

Name of the interface	Name of the function	Function designation
IUCWEComponent	Functions of the interface are described in Table. 2	
IUCWELayer	AddObject	Add new object to the layer
	DeleteObject	Remove the object from the list
	SetCurrentObject	Make the current object on the map
	GetCurrentObject	Return a pointer to the selected object

Component UCWEObject is a game object of the card. Every object is - an entity that accumulates certain properties. The property will be initialized in the editor by the components UCWEProperty. Thus UCWEObject component contains a set of components UCWEProperty. Components UCWEProperty is created by user of the editor in accordance with the plot developed by means of the framework of the game. Initialized components are automatically used in the game. Interfaces of the component UCWEObject are shown in the Table. 6.

Table 6

The list of interfaces of UCWEObject component

Name of the interface	Name of the function	Designation of the function
IUCWEComponent	Interface functions are described in Table. 2	
IUCWEObject	GetPropertiesList	Returns the list of components UCWEProperty as object properties

For each game UCWEProperty components can be own. One of the components UCWEProperty can implement the appearance of the object, the other - the numeric value of the parameter, etc. The component must support a standard interface of IUCWEComponent. It should be noted that each of the components of the editor can support other proprietary interfaces. Especially, this principle regards the components UCWEProperty in the process of implementation of close interaction with the component UCWEEngine, because they are created by means of one game framework. General model of graphical by-component editor is shown in Fig. 3.

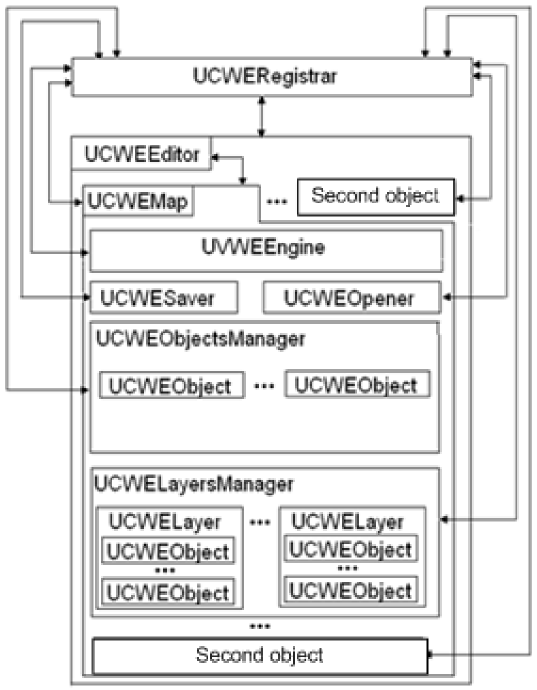

Fig. 3. General model of graphic component editor

Note that under the insertion of one component into another we mean that the first component creates (loads) the last one. Thus, the lifetime of the loaded component is embedded in the lifetime of the component-loader. Components interact with each other by COM technology, realizing the architectural principle of objects management. So, the start before execution of the final file provides loading of UCWERegistrar and UCWEEditor components. UCWEEditor loads directly UCWEMap component and other components, which are listed in the configuration file of component UCWEEditor. UCWEMap, in its turn, loads the corresponding components UCWESaver, UCWEOpener, UCWEEngine. Working file UCWE.exe calls function IUCWEComponent:: Registering component UCWEEditor, which, in turn, calls the same functions in all of its loaded components, encouraging them to take similar actions to the lowest level of embeddiment. Similarly file UZWE.exe invokes IUCWEComponent:: Init. Component UCWEEngine, creating by its own means the window of rendering calls the function IUCWEComponent:: Do component UCWEEditor. The latter provides the call of the same functions in all active components, moving successively to the last level of embediment. If the user selects a menu option of card creation, then UCWEMap additionally loads components UCWEObjectsManager and UCWELayersManager, and cyclic function calls Do are repeated again: When control reaches the function Do of UCWEProrerty components, then, depending on designation, one of them will draw the object, another shades it or graphically realizes certain effect. Thus step by step the creation of objects that form the card is carried out. When the user closes the editor, function UnInit is called and then functions UnRegistering of all components of the editor in the sequence reverse to loading are called. If the user selects the option to save the file, the component UCWEMap calls the function of component UCWESaver saving, providing a chain call of GetData function of all embedded components of the editor and the file saves all data to be provided by these functions. The algorithm of file card opening with the call of component UCWEOpener operates by the same scheme.

Components of the editor can create workflows, so all components should be established taking into account the multi-thread nature. The model the free flow of COM (Free threaded model) is standard for the developed editor.

Using the chain implementation scheme of the architectural principle of objects management ensures reliable operation of the graphic editor and allows to avoid conflicts while data processing.

Conclusion

Universal graphic component editor of game cards, which generalized the capabilities of existing editors, thus providing independence on subjects and genres of games is developed. The problem of independence of the editor independence on specific game carcasses is solved. Due to application of component technology, the editor provides the extension of its functions in the system of open type. Available facilities of card file allow the designers to realize the author's ideas. The editor can be used for modeling of objects systems in various spheres of design and engineering.

REFERENCES

1. Todd Barron. Strategy game programming with DirectX 9.0. – Wordware Publishing Inc.: 2003. – 350 c.
2. Дональд Бокс. Сущность технологии COM. Библиотека программиста. – СПб.: Питер, 2001. – 400 с.
3. Ламот Андре. Программирование игр для Windows. Советы профессионала, 2-е изд.: Пер. с англ. – М.: Издательский дом "Вильямс", 2004. – 880 с.: ил. – Парал. тит. англ.
4. Марк Зальцман. Компьютерные игры: как это делается.: Издательский Дом "Логрус", 2000. – 540 с.

Voitko Victorija – Cand.Sc.(Eng), Assist. Professor, Software Department.

Denysiuk Alla – Assistant.

Lutsyshyn Gennadiy – Graduate Student, Department of Information Technologies and Computer Engineering.

Vinnytsia National Technical University.