System for Interactive Vector Image Processing

Slava Milanova Yordanova [1], Mariana Ts. Stoeva [2]

Abstract: **It is proposed a system for vector image processing – called by the authors "Open Modeler", based of the plug-ins and libraries of the Windows and the object modeler of Microsoft.**
Keywords: **Graphic, Open Modeler, vectors, matrices, images**

I. INTRODUCTION

The standard IGES (International Graphics Exchange Specification) is created in 1979. There are put in the concepts as method of end elements, presentation of areas by interpolation and approximation, splains and others [1, 2, 3]. The company, which ground the standards in this field, is American company Autodesk. Its products as AutoCAD, 3D Studio, 3D Studio MAX are standing as a base for comparing the any other system for vector graphiics. Important significance assumed the visualization libraries Open GL of SGI and DirectX of Microsoft [4].

In this paper a new system is offered for processing of vector graphics - called from authors Open Modeller, which is based on superstructures and libraries of Windows and Object Model of Microsoft.

The purpose of proposed work out is using the contemporary methods for building of engineer vector graphic systems with option for hardware acceleration of visualization and similar interface for computer graphic subsystem access.

II. OPEN MODELLER SYSTEM DESIGN

A. *Main concepts and definitions.*

The Open Modeller program system is build on modular principle. For this purpose is used the Component Object Model of Microsoft.

It is realized on base of interfaces, contains sets of functions, which every one component defined on your creation. For transparency the fundamental concepts will defined connected to the function of Component Object Model [2, 3, 4].

- Component - a program module, which realizes one or more interfaces and eventually defines one or more interfaces. When a component uses the other, it is called client, and the used - server, because it disposes determinate services.
- Interface - a set of functions, which a client can use for calling a component. This function set is declared by mean, so that can be readed by every one, independent of the language and platform on which it works. The interfaces of one or more components are described and saved in so called type libraries, as the operation system give up methods for its registration and readig.
- GUID (Global Unique Identifier) - Each component is identified with a 128-bit number, called GUID (Global Unique Identifier). This number is calculated by a equation when the component is created, the specification of the computer, the date and hour of creation and other factors are taken to insure the uniqueness of these identifiers. The uniqueness is very important, because certain component could be used from many machines simultaneously and its identifiers must not be duplicated with one of other component. There are public databases that keep all registered till now identifiers. They offer secure unique identifiers for minimal cost.
- IID (Interface IDentifier) – When a component defines an interface it defines and GUID. It is possible certain component to define more than one interface so it is necessary to exists a method for their distinguishing. By this GUID each client is shown which component to use. These identifiers are called interface identifiers.
- CLSID (CLaSs IDentifiers) – A certain interface could be realized from different components. To be differentiated, to each of them is given a GUID that is called class identifier. It defines the component that realizes the given interface.
- ProgID (Programatic IDentifier) – They are a sequence of symbols defined by the programmer that give a descriptive name of the component. They are developed to be remembered easily and to be used easily by the corresponding number identifiers. They does not make an uniqueness.

To be identified a certain component is necessary to be known its class identifier(CLSID) and its interface identifier (IID), through which the access to it will be made. Most of the systems have internal support of the access methods. It is necessary to be defined the component that will be used either through the user interface, as in Visual Basic, or through a predefined constants as in Visual C++.

The modularity of the system Open Modeler is granted by the breaking it to two parts. The first and most important part is the core of the system. It is an executable file that controls all the rest modules in the system. Its basic purposes are two:

[1] Slava Milanova Yordanova Technical University Varna 9000 "Computational Technique" Department, Republic of Bulgaria Email:sl@windmail.net
[2] Mariana Ts. Stoeva Technical University, Varna, 9000 "Computational Technique" Department, Republic of Bulgaria E-mail: mstoeva@windmail.net

to load and initialize the rest of the modules and to give them specific services that they use. These services consists in itself common user interface. This consists of common realization of the menus and floating bars with tools. [1, 2].

B. Control and execution of commands defined preliminary by the modules.

For the realization is necessary preliminary to be defined certain definitions that will be used in the development of the system.

- ***Interface element*** – This is each part of the user interface, as the menu, option from the menu, floating bar, etc. There are two kinds of elements like this – local and global.
- ***Local interface element*** – These are interface elements that are visible only when a certain view of a module is active.
- ***Global interface element*** – These are interface elements that are visible all the time. They are used for commands that must be available trough all the working time of the program.
- ***Commands*** – Identifier couple Program identifier – Number that describes a certain element from the user interface or action connected with the user interface.
- ***Event*** – Notification message that is sent to all registered for it modules.
- ***User View*** – A window for visualization.

The second part of the system is complex and consists of all modules that are registered and loaded. The modules are components organized in dynamic libraries (DLL) that are realizing the interface IPlugin and could be two types [1, 2]:

- ***Independent modules*** – These are independent components that are free from the others modules in loading and initialization.
- ***Dependant modules*** – These are components for which initialization are necessary certain preconditions. It means that other modules must be loaded before them to work the system properly. The dependant modules check every time if the modules they need are loaded.

On Fig.1 is shown a block scheme of the modules in the offered system.

Application

It is created first when the program starts and its function is to organize the rest of the components. In the initialization the separate modules receive index, so they can realize theirs own functions, add commands and elements from the user interface and register events. [1, 5]

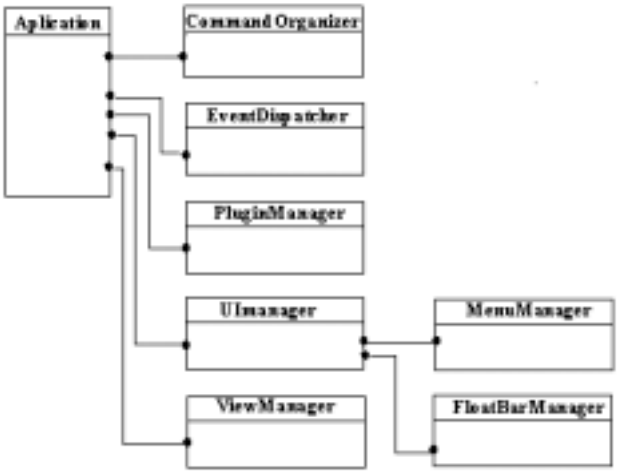

Fig.1 Connections between the modules.

CommandOrganizer

This component is used for organization of the commands in the system. Its purposes are to ensure uniqueness to the commands and to give resources to the modules to register and unregister commands, also to attach itself to definite command for service actions.

EventDispatcher

The basic functions of this component are registration and dispatch of events. These events are synchronous. It means that if an event arrives, it will be sent subsequently to all modules registered for it.

PluginManager

This is a component that is called immediately after creation of the *Application*. Its purpose is to organize all modules registered in the system, to load, to initialize and to unload them. Once activated it shows on the screen no modal dialog that is used for displaying a list of all registered modules in the system and their status – loaded or unloaded.

ViewManager

This component consists and organizes all entered from the modules user views. Trough it every module can append and remove own view that can be unique not only by functionality but by user interface.

MenuManager

It is used for organization of the menus in the system. It is accessible directly from the component *UIManager* and organizes in the same way either the global or the local menus and options.

FloatBarManager

This component organizes the floating panels with instruments their appending and removal. It is used for manipulation of either the local or the global panels and the manipulation of the both kinds is the same. Each panel is identified with Identification Couple. (i.e. Program Identificator - Number)

View

The component defines and controls the user view - for internal use only. In it there is to the local *UIManager* also and the view itself given from the module.

FloatBar

This component is a floating panel. It is used for appending instruments in given panel.

FloatBarTool

It is a component that represents definite instrument from a given panel with instruments.

IPlugin

It is the basic interface that has to realize each module if this module wants to be loaded into the system. Each method of this interface has to be realized from the module.

The visualization of the objects in the system is realized with the help of the visual library Open Cascade 3.0 of the company Matra Datavision Inc. The current version of this library is fully free and is available its source. It has been developed since 1994 from the same company for the purposes of the engineer graphic and consists of function for visualization and geometrical modeling. It is written in C++ and is multi platform. Its structure is object oriented and consists of hierarchy of classes for visualization of plane or spatial objects. For visualization is used OpenGL. This permits to the user of the system to see real time a photorealistic model of his work without the help of other expensive products. In this moment the version 4.0 is in progress that will have better support of hard modeling and also a support of multiple file formats as DXF and DWG.

The main module of the system Open Modeler is realized as independent module that is loaded in the beginning. Its function consists of:

1. Visualization of all objects of the drawing in the space.
2. Drawing the basic 2D and 3D primitives:
   - Line
   - Subline
   - Rectangular
   - Cube spline
   - Regular parallelepiped
   - Etc.
3. Functionality for finding of hotspots on the drawing for improvement of the drawing precision:
   - Definition of the cross points between the objects.
   - Definition of end points of the objects.
   - Definition of point placed on exact distance from the beginning of the frame of reference.

OMD (Open Modeler Drawing) – own file format in binary code.

DXF (Drawing eXchange Format) – text file format from Autodesk.

BREP (Boundary REPresentation) – Internal file format of Open Cascade.

RLE (Run-Length Encoded) – Internal file format of Open Cascade.

III. CONCLUSIONS AND RESULTS.

The program system *Open Modeller* has intuitive interface. Each function is available from the menu and from the floating panels with instruments. In it is used exclusively only the mouse and the function of the it's buttons are clearly separated – the left button of the mouse is used for selection, drawing, modifying and editing of the objects; the right button is used for rejecting of an operation and for finishing of some operations. During the drawing, modifying and manipulating the view it is not possible to be made selection of objects.

For edition of definite object it has to be draw totally and the status has to be in regime selection. The function for modification includes moving, rotation and zooming of the object. For each of them it is necessary first to select the objects that have to be modified.

The program system *Open Modeller* is complete system for vector graphic with capabilities that have all the new programs in this field. The basic composition of functions for drawing, editing and modification of primitives and more complex objects is realized openly. The user interface is easy for understanding and use. The primary advantages of the system are its open architecture, the easy and simple interface, the support of number of file formats and the visualization engine that permits definite object to be viewed in almost realistic form without using expensive software for rendering. The possibilities for extension of the system are infinite. In the common case the system is not limited and can be used in the architecture, constructions and even in the game industry. It is necessary to be written the corresponding modules.

LITERATURE:

[1.] Ibrahim Zeid, CAD/CAM Theory and Practice
[2.] Dan Box, CAM Essentials
[3.] Brent Rector and Chris Sells, ATL Internals
[4.] Stivan Harisan, Computer Graphic – Program Approach