ObVis: A Generic Framework for Information Visualization

Bjoem Heckel[#,+] and Bemd Hamann[+]

[#]IBM Almaden Research Center
San Jose, California

[+]Center for Image Processing and Integrated Computing (CIPIC)
Department of Computer Science
University of California, Davis

ABSTRACT

We present a platform independent, generic and extensible framework for information visualization called ObVis. It allows reusing and customizing existing components (e.g., layout algorithms), supports a variety of data sources (e.g., data base system, files and Web documents) and allows integrating external applications and services. It is decomposed in five different component hierarchies. Recombining and adding components and utilizing the framework enables a rapid development of visual-based applications and visualization tools.

Keywords: Information Visualization, Visual Data Exploration, and Visualization Systems.

1 INTRODUCTION

Nowadays, vast amounts of information are presented to analysts and users of computer systems in general. Traditionally, this information has been presented in textual form. To deal with large quantities of data, visualization techniques (e.g., *cone trees* [1], *glyphs* [2], *magic lens* [3], and *parallel coordinate systems* [4]) have been developed in recent years that allow visual exploration. These techniques are applicable in a wide range of fields. However, most visualization systems are tailored to the application domain and as a consequence, these techniques have been implemented numerous times. Moreover, most of the existing visualization systems primarily focus on representing information. In many domains, it is desirable to perform operations on the underlying data integrating visualization and for example data analysis or process control. This is a crucial issue when it is necessary to transform the original data in order to understand the information content.

In this paper, we describe the design and implementation of a generic and extensible framework for information visualization, called *ObVis*, which allows reusing existing components (e.g., layout algorithms). It supports a variety of data sources (e.g., data base system, files and Web documents) and allows integrating external applications and services. The goal of the ObVis project is to create an infrastructure that allows a rapid development of visualization tools for a diverse range of domains.

The ObVis system is decomposed in five different groups of components that are described in section 2. The interfaces between these component groups are well defined. Recombining existing components stored in a repository supports a fast development of visual-based applications. The different component groups are arranged in a class hierarchy. New components can be added to the component library by extending existing classes. By utilizing inheritance and templates, the extension of ObVis becomes an easy task. The complexity of the ObVis system is hidden in the base classes of the component hierarchies. The basic features can be used by derived classes and be altered to give components an individual behavior. On one hand, this approach allows programmers with little graphics programming experience to create powerful visualization tools without having to know much about the implementation of the underlying technology (e.g., issues like performance optimization). On the other hand, it gives experienced programmers the opportunity to create highly customized applications. ObVis is implemented platform independent to reach a wide audience of users and developers. It supports a variety of data sources including database systems, conventional files and the Internet. ObVis posses real-time capabilities and allows to plug in external applications that - controlled through a visual interface - perform operations on the underlying data.

Addresses: [#]IBM Almaden Research Center, 650 Harry Road, San Jose, California 95120-6099, [+]Department of Computer Science, University of California, Davis, Davis, CA 95616-8562
Email: {heckel, hamann}@cs.ucdavis.edu
Project homepage: http:\\graphics.ucdavis.edu\people\heckel\projects\obvis\index.html

2 COMPONENTS

The ObVis system consists of five component groups: A *domain class hierarchy*, an *information object class hierarchy*, a *layout algorithm class hierarchy*, a *pipe class hierarchy*, and a *services class hierarchy*.

2.1 Domain Objects

The heart of an ObVis application is an instance of a class of the domain hierarchy. This so-called *Domain Object* (*DO*) represents the domain in which the visualization tool is operating. It stores information about the application domain and manages a heterogeneous collection of *Information Objects (IOs)*, called the *Object Collection* (*OC*). The DO also controls the visual representation of the Object Collection and defines the layout algorithms that are permitted to operate on it. It processes events triggered by user interaction or external sources and requests for exporting and importing data through pipes. The root of the Domain Object class hierarchy provides methods to manage the Object Collection (e.g. seek, add and delete IOs), interact with external applications and data sources, and process events. A user communicates with the DO, either directly through its dialog or indirectly through the visual representation of the OC.

2.2 Information Objects

An instance of a class of the information object class hierarchy, called *Information Object* (*IO*), represents an external entity (e.g., a gif file, a row in a relational database or a web documents). Each IO has a class-specific visual representation and a set of applicable operations. A set of basic operations - for example for event processing - are derived from the root class of the IO class hierarchy. IOs have a class dependent primary and secondary function that can be directly activated through the visualization. The primary function usually is the most performed operation on a certain object class. ObVis supports different viewing modes, which differ from each other in the assignment of different primary functions to the object classes. For example in *ExploreMode*, clicking on a picture object displays the picture, while in *RearrangeMode* it cuts it out to let the user place it at a different "place". The secondary function creates a class specific dialog that displays the state of the corresponding IO. It also allows the activation of a set of operations that are applicable to one instance of the particular class (e.g., "segment video" creating a set of keyframe pictures).

2.3 Layout Objects

Layout Objects (LOs) are instances of a class from the *layout algorithm class hierarchy*. A LO is applied to the Object Collection or to a subset of it and spatially arranges the geometric representation of the affected IOs. The state of a LO is displayed in a dialog that is accessible through the DO's dialog. This Dialog is also used to configure the particular layout algorithm. LOs are applicable to a certain IO class and it's descendents. Generic LOs - for example, structural LOs like the cone tree layout - are applicable to objects derived from the root of the IO class hierarchy. Other, more individual LOs require the existence of specific IO attributes and are therefore only applicable to IOs of certain classes.

2.4 Pipe Objects

An instance of a class from the pipe object class hierarchy is called a *pipe object* (*PO*). A pipe exports and imports data to/from external data sources. It acts as a translator, converting the data from the external data format to the internal ObVis representation and vise versa. Pipes can interpret file and web documents, read the structure of a file system or web site, or access databases and digital libraries.

2.5 Service Objects

A *Service Object* (*SO*) is derived from a class of the service hierarchy. It implements an internal operation that can be performed on certain IO classes or it serves as an interface to external applications and processes. SOs have one or more of the following functions:

- *production*: adding objects to Object Collection,
- *consumption*: removing objects from OC,
- *reorganization*: structural alteration of OC, and
- *transformation*: change of attributes of objects in the OC.

3 IMPLEMENTATION

ObVis is implemented in *Java*. As a runtime environment, *Netscape Communicator* is used. For the visual representation of the Object Collection, ObVis utilizes VRML 2.0 [5].

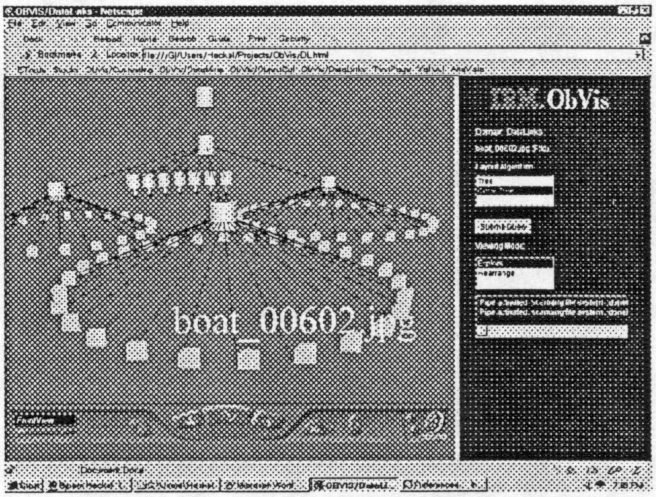

Illustration 1: Screenshot of ObVis/AssetManager.

The user interface consists of two frames in the web browser and additional dialog windows that are created on demand. One frame displays the visual representation of the Object Collection. The other frame shows the GUI of the Domain Object. The communication between the Java Domain Object and the VRML scene is established through the *external-authoring interface*, *EAI* [6]. As a VRML plugin for Netscape Communicator ObVis uses SGI's Cosmoplayer V2.1 [7]. ObVis uses JDBC to directly access a data base or it communicates with external applications via local or remote method invokation. Files and web documents are accessed through the Java *File* and *URL* classes. Currently, ObVis uses an own file format to store and load a Object Collections that is of similar structure as the VRML and OpenInventor file format. In the future, ObVis will utilize the *Extensible Markup Language* (*XML*) [8] for loading and storing ObVis files as well as for the communication with external applications.

4 APPLICATIONS

Currently, the ObVis framework is being used for the development of a variety of visualization-based applications, which are briefly described in the following subsections.

4.1 ObVis/ContentManager

ObVis/ContentManager is being developed to display the state of a content management system consisting of a set of control and service components. It is an administrative tool, which shows the architecture of a content management system. It allows examining the state of individual components and performing operations on them (e.g., start service, and cancel job). Moreover, it displays run-time information (e.g., propagation of objects through the system). Similarly, ObVis could be used to manage other processes, for example production processes or distribution of mail.

4.2 ObVis/AssetManager

ObVis/AssetManager is being designed to interface to an Intranet file asset management system using *DataLinks*. DataLinks is a new IBM database technology [9] that enables management of data external to a database (e.g., files on a file system or web server). DataLinks technology provides referential integrity, access control and coordinated backup and recovery of file assets along with metadata in a relational database. Applications (e.g., web server) can continue to access the files from the file system without change. ObVis/AssetManager will be used to visualize the Intranet site structure and link information, and provide services for version control and for the management and manipulation of data in a DataLinks database. Integrating ObVis with the DataLinks technology will allow managing a large collection of file assets efficiently and robustly.

4.3 ObVis/DocumentExplorer

ObVis/DocumentExplorer is being designed for a visual exploration of large multimedia object collections. It will provide basic operations like searching by metadata (e.g., image name or video description) and browsing. By integrating powerful services, like IBM's Query By Image Content (QBIC), CueVideo and TextMiner, ObVis/DocumentExplorer is expected to become a very useful tool integrating the functionality of different applications.

4.4 ObVis/DataMine

ObVis/DataMine is being designed to explore the result of data mining processes. Currently, it allows the visualization of high-dimensional clusters, and association rules [10]. The utilized layout techniques are *parallel coordinates system* and a new interactive method called *binfields*, which is currently in an experimental stage.

5 SUMMARY

ObVis has proven to be a powerful environment to develop visualization tools and front-ends for information rich applications. With the refinement of the existing applications and the development of new ones, ObVis is expected to evolve further enhancing its capabilities.

6 ACKNOWLEDGEMENTS

This work was supported by the IBM Almaden Research Center and various grants and contracts awarded to the University of California, Davis, including the National Science Foundation under contract ACI 9624034 (CAREER Award), the Office of Naval Research under contract N00014-97-1-0222, the Army Research Office under contract ARO 36598-MA-RIP, the NASA Ames Research Center under contract NAG2-1216, the Lawrence Livermore National Laboratory under contract W-7405-ENG-48 (B335358, B347878), and the Department of Energy as part of the Accelerated Strategic Computing Initiative (ASCI) under contract W-7405-ENG-48. We would like to thank the members of the Visualization Thrust at the Center for Image Processing and Integrated Computing (CIPIC) at the University of California, Davis and the members of the Digital Library research group at the IBM Almaden Research Center.

7 REFERENCES

[1] Robertson, George, Jock Mackinlay, and Stuart Card, "Cone Trees: Animated 3D Visualizations of Hierarchical Information", SIGCHI 91, pp. 189-194.

[2] Jeff Beddow, "Shape Coding of Multi-Dimensional Data on a Microprocessor Display," Proc. Visualization 90, Arie E. Kaufman, ed., IEEE Computer Society Press, Los Alamitos, CA, 1990, pp. 238--246.

[3] Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony D. DeRose, "Toolglass and Magic Lenses: The See-Through Interface," Proceedings of Siggraph 93 (Anaheim, August), Computer Graphics Annual Conference Series, ACM, 1993, pp. 73-80.

[4] Alfred Inselberg and Bernard Dimsdale, "Parallel Coordinates: A Tool for Visualizing Multi-Dimensional Geometry," Proc. Visualization 90, Arie E. Kaufman, ed., IEEE Computer Society Press, Los Alamitos, CA, 1990, pp. 361--375.

[5] Hartman, Jed and Josie Wernecke, "The VRML 2.0 Handbook- Building Moving Worlds on the Web", Silicon Graphics Incorporated, Addison-Wesley Publishing Company, Reading, Massachusetts, 1996.

[6] External Authoring Interface (EAI): http://cosmosoftware.com/developer/moving-worlds/spec/ExternalInterface.html

[7] Cosmoplayer 2.1 VRML plugin: http://cosmosoftware.com/download/player.html

[8] Extensible Markup Language documentation: http://www.w3.org/XML/

[9] DataLinks white paper, available at: http://www.software.ibm.com/data/pubs/papers/datalink.html

[10] Related publications are available at: http://www.almaden.ibm.com/cs/quest/publications.html