Datavis: A research work in Data Management Toolkit

by

Aashish Chaudhary

A thesis submitted to the graduate faculty

In partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Industrial Engineering

Iowa State University

Ames, Iowa

2005

Graduate College
Iowa State University

This is to certify that the master's thesis of

Aashish Chaudhary

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to thank Dr. Carolina Cruz-Neira for her guidance and for her patience. More importantly, I would like to thank her for giving me the opportunity to work with her and her other very talented students at the Virtual Reality Applications Center, one of best research center for Virtual Reality and Human Computer Interaction.

In addition, I would like to thank Dr. Dianne Cook, Dr. Julie Dickerson and Tye Conland from Deere & Company. Their suggestions and goals for the future of data visualization fueled the development of Datavis.

Finally, I would like to thank Jonathan Gurley, Chiou-Guey as my research colleagues, Lew Hill, Levent Engin, Abhishek Seth, and Sailesh Kumar for making VRAC a fun and exciting place to spend most of my waking hours, Shashank Raina, Saikat Adhikari, Anup Gokarn for their suggestions, comments and valuable help during my stay at Iowa State. The environment present at VRAC made all the difference in motivating my efforts toward finishing my master's degree.

ABSTRACT

With increasing popularity of interactive data analysis, simulations and visualization, there is a need for flexibility in the development and use of data management toolkits. Although some work has done by the researchers, a solid foundation is not yet established. This thesis introduces Datavis, a flexible, extensible data management toolkit that integrates simulation systems into data visualization systems. Datavis links together standard numerical simulation tools such as EASY5 [B1] with visualization packages like GGobi making the data flow among these different tools transparent to the user. Datavis allows users to perform data analysis *interactively* independent of the data format or the packages needed in the process. Datavis has three main components: The core that handles the user requests, the data handler which takes care of the organization and storage of the data, and the visualization handler which takes care of the instantiation and manipulation of the visualization system. The proposed toolkit has been successfully implemented and transferred to use as part of routine data analysis activities at a company. At the same time, the work on Datavis has laid the foundation for more sophisticated approaches on integrating simulation and visualization for data analysis.

CHAPTER 1 INTRODUCTION

Defining the Research Problem

As the field of statistical analysis grows, visualization is becoming a critical component of multivariate data analysis. Furthermore with increasing use of databases and specification languages such as XML, data analysts have more flexibility in selection and manipulation of data. Another critical component in data analysis is simulation. In simulation a model is computationally evaluated and data are collected to estimate the desired true characteristics of the model. Data then needs to be visualized and explored. For example, consider a case where a user needs to analyze data from a simulation involving multiple input and output parameters. This data is collected as a result of simulation using some simulation package, which may or may not be in the required data format for the visualization package. The user needs to convert this data into a particular data format as required by the visualization package. The user needs to perform this process every time for any new data from simulation, as there is no live link between the simulation and the visualization package. It gets more complicated if the data needs to be visualized in more than one visualization systems such as GGobi [B2] for desktop and VRJuggler [B3] for CAVE [B4].

As another example, consider a scenario where a user has selected a particular data set for analysis and would like to add some more data or other variables in the same analysis. Typically, this would not be possible with most visualizations systems, as it may require introducing data in formats not supported by the visualization system, or it may require solving a computational model outside the visualization package. In either case, the addition of new data would not be an immediate step during the analysis process. The solution would be to provide a "live" link between the data sources and the interactive system that handles the user request for querying and manipulation of the data.

Most of current visualization toolkits lack in providing a linkage between simulation and visualization systems. Currently, none of the tools available has the ability of moving data between unrelated (or isolated) simulation and visualization packages. For scientists, engineers and analysts the current approach is time consuming and tedious. To get ready for data analysis, these users generally go through the following steps: computing a model numerically; gathering the data; transforming the data in the different required formats for visualization; and then setting parameters manually to perform useful analysis.

It is clear that a more integrated tool that allows the flow of information among the different tools needed in data analysis will significantly enhance the process of data analysis. This thesis focuses on the design and implementation of such a tool, which we have called Datavis. Datavis is a cross platform live link between simulations systems and visualization systems such as GGobi, which is a dynamic graphics program that allows multivariate data to be explored through the manipulation of scatter plots. This link allows data to be passed dynamically and seamlessly between the visualization system and the simulation system. Datavis takes care of all the data format conversions and any other required data management.

The current implementation of the Datavis provides visualization in GGobi but can be extended for other visualization systems such as Matlab [B5]. The data that needs be visualized can be parsed either in text or XML format or in any database management system like MySQL, Oracle. The Datavis has the capability of handling different kind of data types including BLOB[1] data types. The Datavis also has the flexibility to be integrated with any 3D visualization system like VRJuggler API [B6] where one more dimension can be added for better understanding of the data related, for example, with GIS and Bio-Informatics.

Statement of Purpose

The outline of the research work presented here begins with the fundamental requirement of the data management toolkits and packages. Based on these fundamental requirements then we compare current graphics, visualizations, and data management systems. The result of this will be utilized in defining a tool that will enable a truly dynamic data management toolkit for interactive data visualization and data analysis. Datavis, a portable, object-oriented data management toolkit for dynamic data analysis and visualization is the main focus of this research work.

Scope

To fulfill the goal of the research work in the statement of purpose, the work is structured in the following sections:

1. Analysis of requirements for an integrated data management toolkit for simulation and visualization system.
2. Analysis of the existing data management packages.
3. Defining the framework for the proposed toolkit.
4. Initial implementation of the proposed toolkit.

[1] BLOB, binary large object is a data type to store variable size binary data type of any kind such as ASCII text, an executable (*.exe), or a stream of bytes.

5. Iterative work for refinement.
6. Discussions.

A brief description of each section is discussed below:

1. Requirement of the data visualization toolkit (Chapter 1)

   Identify user needs for analyzing data resulting from a simulation. This also includes analyzing user needs for interactive data analysis and areas where user feels limited by the capability of the existing simulation and visualization systems. It would provide a guideline for iterative refinement and goal of the future work.

2. Analysis of the existing data management packages (Chapter 2)

   Survey existing packages to analyze whether already existing packages satisfy uses needs listed above and which ones are missing. Analyzing the existing systems is critical for the research work, as doing so will provide the usefulness and deficiencies of the present system. It is also critical to make sure the goal set for the research project is not met by other existing system.

3. Defining the framework for the proposed toolkit (Chapter 3)

   Define the framework and architecture for the platform independent data management system based on the initial goal set for the research work and outcome of the analysis described above.

4. Initial implementation of the proposed toolkit (Chapter 3, Chapter 4)

   Provide initial implementation for the proposed toolkit, which would be simplified. This would be used as a proof of concept.

5. Iterative work for refinement (Chapter 5)

   Refinement of the initial implementation based on the feedbacks from users at Iowa State University and John Deere, Dubuque. Incorporate suggestions and feedback to make the API more useful and efficient.

6. Discussion (Chapter 6)

   Explore benefits and problems with the Datavis by using it in the existing applications. Define the future work on the API.

CHAPTER 2 REQUIREMENTS OF A DATA MANAGEMENT TOOLKIT

Before proceeding further into requirements it would be appropriate to define the keyword *data*. According to the definition given by Dr. Cook [B7] data is defined as:

"Information that is structured in some schematic form such as a table or a list"

Data can be raw or processed. Data collected might be gathered from the results of simulations, from an agricultural field, from a chemical reaction or from general surveys to mention a few sources. With this definition we present the components required for an integrated data management system that allows data to flow between the data repository or numerical sources and the visualization system. The minimum components required for such an integrated data management system is as follows:

- Efficient data transfer between simulation systems and visualization systems
- Support for heterogeneous visualization systems
- User interface
- State management

This chapter describes each component in detail, and its resource needs or impact on the overall system.

Efficient Data Transfer

This is one of the most important components for a data management system. The usefulness of any data management system depends upon this factor. The data transfer from any data source to the visualization system should be quick as the selected data generated from the simulation system or stored in a database can be quite large. Special care needs to be taken when transferring the data to the visualization system as different visualization systems have different ways to present the data to the user.

Support for Heterogeneous Visualization Systems

There are many data visualization systems available (Open Source and Commercial) such as GGobi, webFocus from Information Builders Inc. [B8], OpenDX from IBM [B9] and PartekPro from Partek Inc. [B10]. Depending upon needs and preferences, user may choose a particular onset of visualization systems for their data analysis activities. An integrated data management system should be able to be integrated to these different systems seamlessly with least effort form the developer point of view.

User Interface

User interface is now a necessity that makes any software so much easier to use especially for the common users of the software. User interface provides easy access to the different functionality of any software. It is very much required for better interaction with the data as it gives users the capability of selecting / filtering the data with certain parameters before it can be visualized.

State Management

State management is required for the data visualization system that links with the data management toolkit. The capabilities for querying state variables and the state of a particular region within visualization plots is necessary for the interaction. This would also allow the user to set any state or mode for a particular instance of data visualization system.

CHAPTER 3 EXISTING DATA VISUALIZATION PACKAGES

Before the implementation of Datavis is underway, various popular existing commercial and open source dynamic data visualization tools were researched and evaluated. This has been performed so as to assess present tools and packages in terms of their technology, drawbacks and advantages. This chapter describes most commonly used and popular packages.

Before describing details on various packages, a short of description of GGobi is presented as most of the discussions are based on an understanding of GGobi. GGobi is a data visualization system for visualization of high-dimensional data. It provides various interactive and dynamic methods for the manipulation of views of data. It also provides the capabilities such as direct manipulation, dynamic visualization functionality, which can be embedded with in other applications.

R-GGobi API

R is an environment and language for statistical computing and graphics [B11]. It is a continuation project from GNU. R provides a wide variety of statistical tools and techniques. In R-GGobi the GGobi is embedded in R and expressed as R-GGobi package [B12]. The GGobi developers develop this API.

Description

R-GGobi is an excellent tool for dealing with GGobi as it provides facility for query and modifies the state of GGobi. Once can initiate GGobi with or without initial data and then can add the data afterwards. The R-GGobi API does not provide any interface for dealing with GGobi as it provides only the console for initiating, query and modify the GGobi instance.

Strengths

As stated above R-GGobi is an excellent tool for initiating GGobi from the R, set the data and query current state. It also provides limited support for using R functions implementing some of the GGobi user-specifiable facilities. R-GGobi also provides event handling so as to update the appearance for GGobi. It provides the user the ability of programmatic display generation and non-standard layouts.

Limitations

The focus of developers of R-GGobi API is only towards the statistical language R. This API has very strong binding with R, which puts a limit to the user as then the user is forced to use R, which may or may not be suitable for user requirements. Another shortcoming of R-GGobi package is that it does not has capability of linking with the simulation systems and packages.

Arc-GGobi

ArcView [B13] is an excellent geographical information system that provides the user ability to display, query and manipulate spatial data. ArcView-XGobi [B14] links provides the facility to the user to use the power of both tools.

Description

This Arc-GGobi API came into existence so as to provide the facility of transferring the spatial data dynamically to the GGobi. The link between the data points and the location from which those data point were collected is maintained through some mechanism so that when a particular set of data points are selected then corresponding geographic locations will be highlighted in the map.

Strengths

This API is successfully implemented on SGI and Sun / Solaris workstations. The API is very useful for the users having large database of spatial data which they intended to visualize in GGobi.

Limitations

Again the focus of the developers is towards ArcView linkage with GGobi only and it was not intended to provide any link with any simulation systems and packages. Also this project is not under any current development.

Data Visualization Toolbox for MATLAB

"Data Visualization Toolbox" is a tool for data analysis and visualization from Datatool Inc. [B15], a consulting service on Matlab related products. This software is free for academic or non-profit use.

Description

It can be best described as MATLAB software [B16] that implements the analytical and graphical methods for data analysis and visualization. It is primarily based on methods described in the William S. Cleveland's *Visualizing Data* [B17].

Strengths

"Data Visualization Toolbox" provides analysis using robust statistical methods, graphical methods for data exploration and diagnostics for fit and model evaluation.

Limitations

This toolbox is intended for the Matlab users only. The toolbox provides limited functionality and does not have the ability to link with the simulations systems.

Vis5d+

Vis5d+ [B18] is a software system that can be used for visualizing data in a 3D grid. It is based on Vis5d [B19], an OpenGL-based program for visualizing scientific datasets in three dimensions.

Description

Vis5d+ is enhanced version and development work on Vis5d, a system for interactive visualizations of five dimensional gridded data. Vis5d was written at University of

Wisconsin, Madison with the support of National Aeronautics and Space Exploration (NASA) and the Environmental Protection Agency (EPA). Vis5d+ works on most of the operating systems such as Linux and IRIX.

It has the ability to work on the five dimensional data. The grid represents the three space dimensions, one time dimension and one for enumerating physical variables. One of the important features of the Vis5d+ is the ability to compare multiple data sets. The data can be loaded at run time, which then can be overlaid in 3D display or a spreadsheet format.

Strengths

Vis5d+ includes number of utility programs such as v5dinfo and v5dstats that would priced the information the data file such as the size of the 3-D grid, the names of variables and some simple statistical information about the grid data. It is possible to extend Vis5D that would run in Virtual Reality and Cave environments.

Limitations

Vis5d+ accepts data in only two standard file formats. Though the user can create a new file format but then that would require the user to change the source code of Vis5d+ or write some conversion program. Additionally Vis5d+ does not have the ability to be linked with simulations and other visualization systems.

OpenViz

OpenViz [B20] is a commercial data visualization package from Advance Visual Systems (AVS). OpenViz provides Web-based and desktop analysis, visualization and performance monitoring.

Description

OpenViz is a powerful tool that provides the user the ability to have 2D / 3D interactive visualization presentations that can be easily distributed via the Web and added to new or existing application. OpenViz has the ability to deal with any type and size of the data. This software package is platform independent in the sense that is provides OpenViz for COM (Windows XP) and OpenViz for Java (PCs, Macs, UNIX, and Linux). It is customizable and provides text, legends and brushing.

Strengths

OpenViz enables any type and quantity of the data to be presented in two and three-dimensional. It is highly customizable, portable and scalable. It provides the capability for distributing the 2D/3D presentation over the Web or adding it to the new or existing applications.

Limitations

Though OpenViz is highly capable for the data visualization it does not fulfill the requirement for a data management toolkit, as it cannot be linked with simulation packages. It's completely a data visualization system. Also it cannot be extended for the Virtual Reality environment.

Summary

Based on the research and evaluation of the mentioned packages we have found that most of the systems do not have the ability to be able to tie up with the simulation system. Only few have the ability to deal with multiple data types like OpenViz. Most of the tools are strongly bonded with a particular API or a software package like R-GGobi and Data Visualization Toolbox. Based on our comparison of these popular packages we are now presenting the design of Datavis in Chapter 4 which incorporates all the requirements for a true data management toolkit along with all the features missing in these packages.

CHAPTER 4 ARCHITECTURE AND DESIGN OF DATAVIS

Based on the study into the visualization packages as described in the previous chapter, we felt the necessity and requirement of a new implementation that would incorporate the most useful features from the studied systems along with the new features that were missing.

Some systems are designed for specialized purposes such as ARC-GGobi and R-GGobi. Others are tied to a particular operating system or a particular visualization system. None of the above implementations has the ability to be linked with the simulations systems.

To overcome these limitations, Datavis has the following main goals:

- Independence from the visualization software
- Independence from the simulations software
- Efficient data transfer
- Ability to handle different data types
- Easy addition to the existing data visualization applications

The following section describes the API and the motivation behind the design. Next section will describe each component in detail.

API Structure

Data management toolkit is divided in three parts, the core library, the general data visualization interface and the general database interface. The core library handles the events, communications and objects. The general data visualization interface provides the interface used by all the visualization implementations and the data specific interface provides the interface for handling multiple data types. Toolkit uses the object-oriented (OO) programming paradigm.

C++ programming language is been used to make this toolkit work on heterogeneous computing environment. Extra care has been taken not to use any platform specific calls that would bind it to a particular operating system. Toolkit takes advantage of inheritance to encourage code reuse. Use of object oriented design benefits programmers in many ways. The implementation is hidden from the programmer and it allows them to think in terms of data. Data management toolkit doesn't visualize/render the data by its own but relies on some other packages / systems to provide visualization for the data. This makes it possible to add Datavis to existing application.

Core Library

The core of the Datavis takes care of the object creation and handling user inputs. The core also provides wrapper for the functionalities provided by modules of the Datavis. As stated before, Datavis uses C++ as programming language and any application, which can interface with C++ libraries, would be able to use Datavis. To be able to use this library in Python programming language Datavis is using the exceptionally well written library Boost.Python [B21], which provides quick and seamless exposure of C++ class functions and objects to Python and vice versa. Python is an interpreted, interactive object oriented language, which incorporates modules, classes, functions and various data types. The best advantage of using Python is its extensibility in C++ and portability on multiple operating systems. Core of

Datavis provides wrapper around modules in C++ so that the objects in Python can access functionality provided by modules of Datavis.

General Data Visualization API

This API provides a general interface for the programmer who would like to construct their own data visualization system specific implementation as long as it conforms to the interface set forth by Datavis. Here the programmer would make calls to a specific data visualization library. Datavis itself provides provide GGobi specific data visualization implementation as an example for using general data visualization API.

General Database API

Visualizations are emerging as a very important part of databases. This is an emerging topic in the database community [B22]. Also databases are increasingly gaining importance in the field of visualization. Previously researchers and scientists acknowledged the importance of databases in visualization [B23]. Most of the visualization systems has poor data management systems and provides very basic data querying and/or file browsing capabilities. By providing linkage with databases in Datavis, we are providing all the capabilities offered by a particular database system for effective browsing and storage tool so that user can easily navigate and interact with the information generated by the simulation systems. A programmer has to implement this API for a particular database system interface.

CHAPTER 5 DETAILED DESCRIPTION OF DATAVIS

Based on the background presented in the previous chapters we now present the implementation of Datavis. We begin by describing the core components of Datavis followed by general data visualization API and database API.

Core Component Structure

This section presents the structure of core components of Datavis. As stated earlier, core of Datavis exposes C++ functions to Python using Boost.Python. Boost.Python is exceptionally well written library and a framework for interfacing Python and C++. It is designed to wrap C++ interfaces non-intrusively, so that the user does not have to change the C++ code in order to wrap it. This makes Boost.Python ideal for exposing 3rd-party libraries to Python. Boost.Python uses advanced metaprogramming techniques, which simplifies its syntax for users.

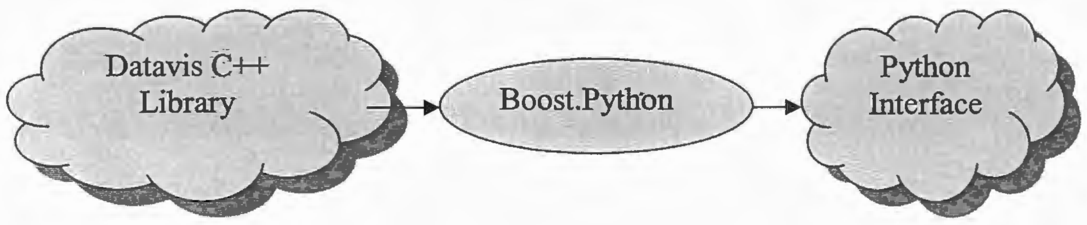

Figure 1 Core component structure layout

To show how functions in C++ library are exposed to python, an example is given below where function `setPlotType (std::string)` defined in class `GGobiPlotDisplay` is exposed to Python.

```
struct dataview_GGobiPlotDisplay_Wrapper:dataview::GGobiPlotDisplay
{
      dataview_GGobiPlotDisplay_Wrapper
            (PyObject* self_, const dataview::GGobiPlotDisplay &
             p0):dataview::GGobiPlotDisplay (p0), self (self_) {}

      dataview_GGobiPlotDisplay_Wrapper
            (PyObject* self_):dataview::GGobiPlotDisplay (), self
            (self_) {}

      void setPlotType(const std::string& p0)
      {
            call_method< void > (self, "setPlotType", p0);
      }

      void default_setPlotType(const std::string& p0)
      {
            dataview::GGobiPlotDisplay::setPlotType(p0);
      }
}
```

Figure 2 C++ application code example

User has to create an instance of `GGobiPlotDisplay`, a GGobi specific implementation of general data visualization API, in python as,

```
      self.plot = GGobiPlotDisplay ()
```

And to generate a particular plot type pass a string of predefine type as,

```
      self.plot.setPlotType (plotType)
```

Basically user has to get an instance of the class `GGobiPlotDisplay` in the program and invoke functions as required. The same can be done for an interface written in C++ or any other programming language, based on the condition that it can use the functionality provided by software libraries. The core maps functions with their respective classes. If the user is having a graphical user interface for accessing API calls then it would be the user's responsibility for separating the thread for the interface.

Below is the diagram showing how Datavis can be connected to various components. The data visualization interface makes appropriate function calls for initialization, gathering the data and making the visualization system interface visible to the user.

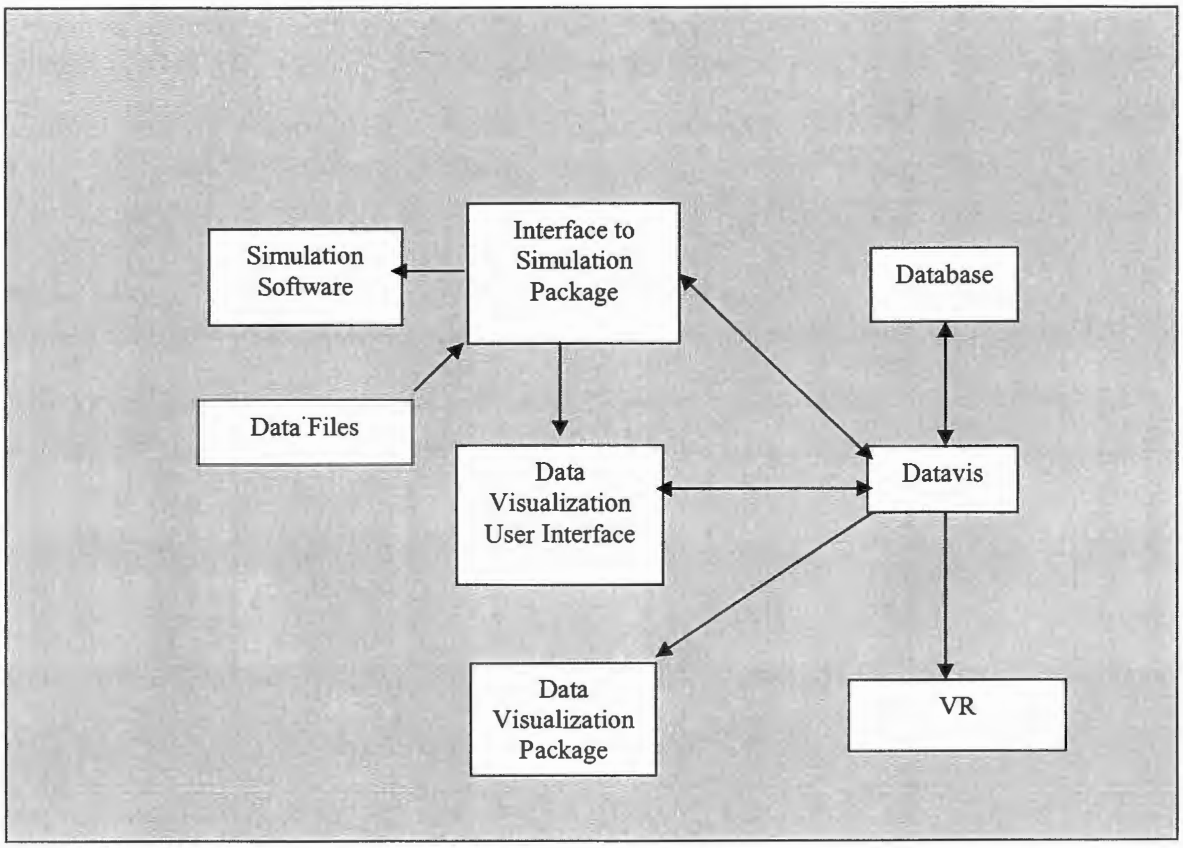
Simulation Software
Interface to Simulation Package
Database
Data Files
Data Visualization User Interface
Datavis
Data Visualization Package
VR

Figure 3 Datavis interaction layout

Interface to the simulation package start the simulation with some selected parameters, which generates data files with output data values. These files may be saved to the database, which then can be queried by visualization interface via Datavis database API. Once the data is selected, Datavis makes appropriate check for the data type and then pass it to the visualization system.

General Data Visualization API

This interface is designed for ease in use and is extensible. At its most general level, the interface defines several useful methods and member variables common to all the implementation of this API. Any implementation derives from general data visualization API must implement its own `displayPlot ()` that defines how to draw different kinds of plot on the screen and `hidePlot ()`, which hides the plot that's already drawn on the screen. All other functions can be overridden as necessary. This gives user an ability to display the data only when it's desired. Nothing will be drawn unless user would make this function call. This concept is similar to the "*DisplayLists*" in OpenGL [B]. The interface has state variables `mVisible` and `mType` that stores the current information whether the data is visible to the user and the information for the current plot type respectively. Once these state variables are set they would remain the same unless changed by the user.

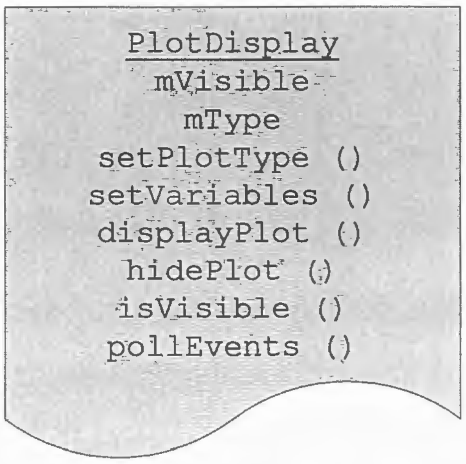

Figure 4 Data visualization API structure

Implementation

Datavis provides implementation for the General Data Visualization API for GGobi. This simplifies the users work and provides an example on implementation of the

API. `GGobiPlotDisplay` is the class derived directly from `PlotDisplay` and defines its own draw () function. This class also adds its own functions to provide extra functionalities required only by GGobi. The data is not directly sent to the setVariables () of `PlotDisplay` but passed with in the function `setVariablesForC` () which takes care of the appropriate data format expected by `setVariables` (), a Boost.Python shared pointer on standard C++ library map (a map of column names to their values).

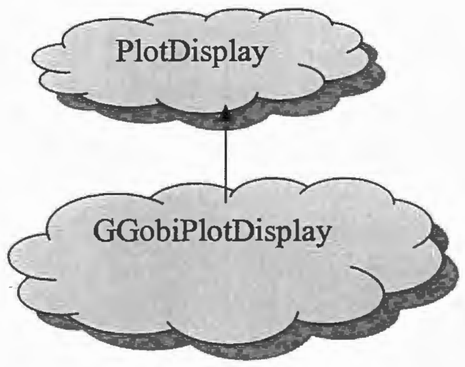

Figure 5 Data visualization API class structure

General Database API

General database API provides the necessary functionality to store, retrieve and query the data generated by simulation packages. This provides an extensive advantage to the user as described in Chapter 5. The API is implementation independent and contains common data members and methods required by all the database system specific implementations. API contains an abstract base class "Database" from which the database system specific sub classes are derived. Subclasses have to provide its own implementation for the virtual functions declared in the base class. The API contains data members such as `mDBName`, `mUser`, `mPassword`, and `mHost`, which basically are required to connect to the database server. State variables such as `bQuery` and `mCurrentTable` store the current state of query state and current table being queried. The end user does not to deal with this API directly but with the implementation of this API.

Implementation

Datavis provides implementation of the General Database API for MySQL database systems. MySQL-C API is used to provide access to the database and all other functionality facilitated by the MySQL database systems.

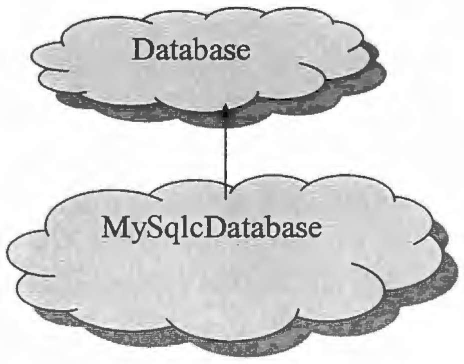

Figure 6 Database API class structure

CHAPTER 7 DISCUSSION

In Chapter 4 the goals of Datavis are presented. Most of the goals were achieved by careful design and redesign of the Datavis. Datavis has achieved independence from any of the simulation and data visualization systems and provides seamless integration between them. It has the ability to handle different data types and provides an API for the databases so that the visualization can be benefited from the capabilities of the database systems.

As mentioned earlier that Datavis primarily uses object-oriented technology and therefore expects that the user has some background with C++ and object-oriented technology. To integrate Datavis with the current application does not require the user to redesign the application from the object orient prospectus. This makes it easier to use Datavis with the existing applications. Though providing an implementation of the General Data Visualization API is straightforward but then registering this implementation with core could be the difficult part. The core of the Datavis uses Boost.Python library and hence user would require learning some basics on Boost and Boost.Python. Theoretically it would have been possible to provide some solution to this problem and needs to be investigated.

CHAPTER 8 FUTURE WORK

Datavis is an evolving project and based on the users feedback and project goals several limitations of Datavis has been recognized. These limitations are mentioned below:

Self Registration of the Implementations

As stated in chapter 7, the implementation has to be registered in the core of the Datavis and that requires the user to have some knowledge of the Boost libraries. Even though Boost libraries are so powerful, understanding and using them would be difficult for certain users. These complexities needs to hidden from the user to achieve the goal of making Datavis easy for the user to use in existing applications.

Multi-Threaded Capabilities

Currently it is the user's responsibility to create and manage multiple threads if required for the visualization system's graphical user interfaces. It could have been done on the API side so as to make it easier to use Datavis. Possibly it can be done by creating a separate thread for displaying the data that would maintain the loop for the graphical user interfaces.

Incorporating Additional Data Visualization Techniques and Methods

Datavis can generate different plot types as requested by the user. Complete data visualization software might have more than just drawing different plots on the screen. This area needs to be investigated and researched.

BIBLIOGRAPHY

[B1]. EASY5, Simulation and Analysis Toolkit, *http://www.mscsoftware.com*, as visited on 04/15/2005.

[B2]. Symanzik, J., Swayne, D. F., Temple, D.L. and Cook, D. Software Integration for Multivariate Exploratory Spatial Data Analysis. *In Proceedings of the Specialist Meeting, Santa Barbara, California* (May 2002).

[B3]. Cruz-Neira, C., Bierbaum, A., Hartling, P., Just, C. and Meinert, K. VR Juggler–An Open Source Platform for Virtual Reality Applications. *In 40th AIAA Aerospace Sciences Meeting and Exhibit 2002, Reno, Nevada*, AIAA Paper 2002-0754 (January 14–17 2002).

[B4]. Cruz-Neira, C., Sandin, D.J., DeFanti, T.A. Surround-Screen Projection-Based Virtual Reality: The design and Implementation of the CAVE. *In Computer Graphics (Proceedings of SIGGRAPH '93), ACM SIGGRAPH*, p. 135-142, August 1993.

[B5]. Matlab, A High Level Technical Language, *http://www.mathworks.com/products/matlab*, as visited on 04/15/2005.

[B6]. VRJuggler API, *http://www.vrjuggler.org*, as visited on 04/15/205.

[B7]. GGobi Data Visualization System, *http://www.ggobi.org*, as visited on 04/15/2005.

[B8]. webFocus Visual Recovery, Information Builders Inc., *http://www.informationbuilders.com/products/webfocus/data_visualization.html*, as visited on 04/15/2005.

[B9]. OpenDX, Open Visualization Data Explorer, *http://www.research.ibm.com/dx*, as visited on 04/15/2005.

[B10]. PartekPro, interactive visualization system, *http://www.partek.com/html/products/pdf/Pro-Features-v60.pdf*, as visited on 04/15/2005.

[B11]. R, Software Environment for Statistical Computing and Graphics, *http://www.r-project.org/index.htm*, as visited on 04/15/2005.

[B12]. Temple Lang, D. and Swayne Deborah F. GGobi meets R: an extensible environment for interactive dynamic data visualization. *Proceedings of the 2nd International Workshop on Distributed Statistical Computing March 15-17, Vienna, Austria (*2001).

[B13]. ArcView, GIS Software, *http://www.esri.com/software/arcgis/arcview*, as visited on 04/15/2005.

[B14]. Cook, D., Symanzik, J., Majure, J. J., Cressie, N. (1997): Dynamic Graphics in a GIS: More Examples Using Linked Software, *Computers & Geosciences: Special Issue on Exploratory Cartographic Visualization*, Vol. 23, No. 4, 371-385.

[B15]. Datatool Consulting Services, *http://www.datatool.com*, as visited on 04/15/2005.

[B16]. Matlab software, *http://www.mathworks.com/products/matlab/description1.html*, as visited on 04/15/2005.

[B17]. Cleveland W., *"Visualizing Data"*, AT&T Bell Laboratories, March 1993.

[B18]. Vis5d+, OpenGL-based volumetric visualization program, *http://vis5d.sourceforge.net*, as visited on 04/15/2005.

[B19]. Vis5d, Interactive Visualization System, *http://www.ssec.wisc.edu/~billh/vis5d.html*, as visited on 04/15/2005.

[B20]. OpenViz White Papers, 2004 Advanced Visual Systems Inc., *http://www.avs.com/download.html*, as visited on 04/15/2005.

[B21]. Boost.Python C++ library, *http://www.boost.org/libs/python/doc*, as visited on 04/15/2005.

[B22]. CLOUDS: Online Database Visualization, *http://datasplash.cs.berkeley.edu/online*, as visited on 04/15/2005.

[B23]. Kochevar, P., Ahmed, A., Shade, J., and Sharp, C. Bridging the gap between visualization and data management: A simple visualization management system. In *Proceedings Visualization '93* (San Jose, California, 1993), pp. 94-101.

[B24]. OpenGL API, http://www.opengl.org, As visited on 04/15/2005