On-Line and Off-Line Multi-Dimensional Data Presentation

S. A. Kryukov
Institute for Nuclear Research, 60-th October Anniversary prospect, 7A, Moscow*, 117312 Russia*

Abstract

A new WIN32 version of the MOON Lab, a program system for development and running data acquisition and control applications, introduces new capabilities for support and presentation of multi-dimensional data. The system has been developed at the Moscow Meson Factory Linac.

The paper covers description of the object-oriented program interface for support of multi-dimensional data by the MOON Lab client process (User Process). The interface is abstracted from the file format. The file resulting from the collecting of data with use of the program interface can be considered as an exhaustive report for entire experiment. The related data structures encapsulate virtually every aspect of the experiment, including date, time, duration of data acquisition, structure of dependencies, comments, etc., so the file can be passed from one researcher to another with virtually no additional explanations.

A universal data viewer enables to walk through the multi-dimensional spaces and view a function on any 1-dimensional or 2-dimensional subspace as a 2-dimensional or 3-dimensional plot respectively.

Besides, for the MOON Lab client process a program interface for on-line plotting of data is provided. The plot windows and plot elements are created and modified by the client process with use of a special presentation server. Different types of presentation are discussed and compared. Also discussed is the choice between on-line plotting of data and collecting of data in the MOON Lab multi-dimensional data file for future research. Both possibilities are considered as necessary to achieve both run-time flexibility and consistency of the experimental data.

Keywords: application programming, data acquisition systems, data presentation, GUI's, object-oriented methods

1. Introduction

MOON Lab, originally projected mainly for solving problems for beam diagnostics, is an environment for running data acquisition and control applications on a local area network or a single computer [1], [2]. These MOON Lab-specific applications can be developed with an easy-to-use interface library supplied. Version 1.0 of this software is presently available for Microsoft® Windows™ and all compatible versions for IBM PCs. Major part of the code has been ported or rewritten for 32-bit Windows 95 and Windows NT® systems.

The MOON Lab versions 2.0 is developed using WIN32 API (Application Program Interface). It can be run on Windows 95 or NT (and cannot be run on Windows 3.* with WIN32s). Program architecture of this version differs significantly from version 1.0, it is easier for developers of applications and provides better performance. On-Line graphics provides virtually the same capabilities than that of version 1.0, and off-line multi-dimensional data presentation has been added.

Main goal of the MOON Lab design was to exclude direct use of the operational system program interface by control or data acquisition applications. This should simplify application programming, make it system-independent and also improve run-time reusability of program components.

Application programming technique using MOON Lab is neither event-oriented nor object-oriented but it is based on simple sequential coding. From the other hand, the behavior of the application produced involves strong event-oriented features because of underlying nature of any elaborated experiment. To couple event-driven structure of application with sequential code developed by the application programmer some multi-tasking and multi-threading approach is essentially used [2].

In this approach MOON Lab plays the role of intermediate-level operational system for running of MOON Lab-specific sequential applications, each application representing a plan or scenario of an independent experiment. All the event-oriented and multi-threading features of the application developed are hidden from the application programmer in the code of MOON Lab system.

One of the most important parts of MOON Lab is a Graphical User Interface (GUI) server, which provides a graphical shell (referred to as the ***MOON Lab Shell***) for each MOON Lab graphical application (referred to as the ***User Process***). MOON Lab Shell serves for on-line presentation of data prepared by the User Process. It can also be used for off-line graphical analysis and presentation of multi-dimensional data obtained from current or one of previous experiments. User process is linked using the API library, referred to as the ***MOON Lab API***.

The MOON Lab GUI server is not a pure *presentation server*, because it additionally provides a possibility of asynchronous modification of application's behaviour and parameters controlled by the user (***Asynchronous Process Control*** [2]). This aspect is out of reach of present considerations.

2. Program Architecture

Present discussion of program architecture is restricted by the 32-bit version 2.0 of MOON Lab. Paper [2] contains more in-depth analysis of program architectures of both versions.

All the MOON Lab API can be used by applications via the *Dynamic Link Libraries* (*DLL*), each function can be called by virtually any procedural programming language.

2.1 Application Programming Technique

The developer of the User Process has to generate an executable module. Major part of the code is not called directly. Instead, this module must *export* at least one function, named ***UserProcess***, representing the body of the process and containing all the calls to the MOON Lab API. Additionally can be exported ***UserConstructor*** and ***UserDestructor*** and some other MOON Lab-specific *callback* functions. Main procedure of the executable module of the User Process must contain only one call to the API, called ***AttachToMOONLab***.

The MOON Lab system either creates a new thread running the MOON Lab Shell or (optionally) attaches User Process to existing *instance* of the MOON Lab Shell, running as a separate *process*. The function UserProcess is run concurrently in the thread separate from the GUI, affecting graphics by using of messaging mechanism.

2.2 On-Line Presentation of Data

The MOON Lab API for on-line data presentation is simple to use, yet powerful.

The MOON Lab Shell design is based on object-oriented hierarchy of graphical *classes*. At the level of the application programming the style of creation and usage of graphical objects is object-oriented, but the inheritance or late binding, however, is impossible; graphical objects are represented by their *handles*; in this way methods of objects became callable with use of non-OOP languages, too.

On the top level of the GUI the three types *Plot Windows* can be created: for 2-dimensional plots, for 3-dimensional surfaces and for 3-dimensional surfaces presented as 2-dimensional color map.

The 2-dimensional Plot Window contains three lists of graphical objects: *plots*, *histograms* and a heterogeneous list of *labels* of different types. Each plot can contain a set of points, represented by pair of mathematical coordinates; number of points is not fixed; any point can be added or deleted at any time by the User Process; coordinate of each plot can also be modified at any time; besides, any *property* of any existing

plot (colour, fill colour, style, etc.) can also be modified at any time; all the run-time modifications is updated on screen by causing invalidation for appropriate subset of window points. To make updates of graphics smooth special buffering technique is developed. *Histogram* is a simplified form of plot, developed to implement fixed number (array) of points with faster procedure of coordinate modification; it can accept styles looking exactly identical to analogous styles of plot. List of *labels* can contain *text*, *bitmap* and *icon* images of different styles, with coordinates attached to element of frames, physical coordinates relative to the window or mathematical coordinates in the coordinate system currently in use.

The 3-dimensional surfaces also support run-time modification, but dimensions are fixed at the moment of creation.

2.3 Multi-Threading Nature of On-Line Presentation

Most important feature involved in the on-line graphics is multi-threading nature of the presentation. The on-screen appearance of graphics is the result of behaviour of two different threads: one representing actions of the user via the GUI, another one related to the code of the User Process. Graphical objects shared by both threads are accessed with use of *Critical Section* and *Mutex* (*Mutual Exclusion*) objects.

Due to different roles of the GUI user and the User Process some sort of the *document-view* approach is developed.

The User Process thread supplies a *document* part of graphics: graphical data itself and properties of plot objects, it also specifies initial and default coordinate system (defined by limits of axes), captions for window and axes. It does not specify many aspects of particular presentation of plots: sizes or position of the plot window; it can not control current parameters of the coordinate system used for plotting data; plot data are supplied in mathematical coordinates only, so the User Process is not aware of the region of plot actually visible on screen.

The *GUI* thread determines a *view* part, that gives a resulting on-screen appearance of graphics supplied by the User Proceess thread, taking into account current sizes and position of the window. The user can modify most properties of graphics during run time via the GUI, including system coordinates (axes limits), optionally using drag-and-drop technique, gridlines for each coordinate, etc. From the other hand, some actions are prohibited for the user; for example, he can not close the Plot Windows when some User Process is attached to the instance of the MOON Lab Shell (the window is minimized instead).

Note, that in contrast to most graphical system most aspects of graphics are neither specified during design-time no during run-time but determined automatically. For example, nobody can specify number, style, sizes and numeric format of numeric labels on axes. These are recalculated during run-time each time the size of the Plot Window or axes limits are modified. Besides, creation of the graphical object does not involve specification of particular style or other property, the default settings are used. Instead, all the properties can be dynamically modified during run-time when it is necessary. The main goal was not to request any redundant information from the application programmer or the user.

2.4 Off-Line Presentation of Multi-Dimensional Data

The MOON API contains add-on API set for support of creation of multi-dimensional data files. It can be done by either User Process or any other application (for example, intended for post-processing of experimental data). The related data structure is supported internally by appropriate MOON Lab DLL and abstracted from particular presentation of data in the form of file.

The structure represents a set of multi-dimensional arrays of floating-point numbers, called ***Functions***. Space dimension of the *Function* (*rank*) and dimension of the array in each dimension are defined by list of dependencies, which is a list of indexes, given from the list of ***Arguments***, representing single-dimensional array of floating-point numbers. During the process of forming the data structure dimension of any *Argument* array and content of the dependency list can be modified at any time.

Schematic structure of the multi-dimensional data:
General Attributes

List of Arguments:
    Number of Arguments = A
    Argument 0:
        Variable Attributes
        Number of Points = N
        x_0, y_0, x_1, y_1, ..., x_{N-1}, y_{N-1}
    Argument 1: ...
    ...
    Argument A-1: ...
List of Functions:
    Number of Functions = F
    Function 0:
        Variable Attributes
        Number of Dependencies = N
        Dependency = Dependency_Index$_0$, Dependency_Index$_1$, ..., Dependency _Index$_{N-1}$
    Function 1: ...
    ...
    Function F-1: ...;

where Dependency_Index is in range 0 to A-1; General Attributes include file name, data name, caption and comment, date-time attributes, duration of experiment; Variable Attributes include variable name (string identifier), axis caption and legend, and also *current index* (multi-dimensional for functions).

Note that different functions can share the same arguments; one-to-one correspondence of function values and argument values is parametric, based on values of indexes.

A universal data viewer enables to walk through the multi-dimensional spaces and view a function on any 1-dimensional or 2-dimensional subspace as a 2-dimensional or 3-dimensional plot respectively. For presentation of plots the MOON Lab design uses that software components (classes) used for on-line data presentation. One can copy plots from the multi-dimensional viewer to another *Plot Window* via the clipboard. The data can also be placed in the system clipboard in the text form; usually this is used to place data in a speadsheet.

3. Discussion

By our experience 3-dimensional graphics can rarely serve as an exhaustive presentation of experimental data. Many experimental processes can be readily presented using sequence of 2-dimensional plots, but other ones produce data presented as multi-dimensional. For that sort of computer-aided experiments collecting data in a multi-dimensional database convenient for graphical off-line analysis must be considered as very desirable.

For on-line graphical presentation, however, 2-dimensional graphics is most useful. Anyway, the possibility of choice between on-line and off-line presentation of data based on the same software components seems to be very important.

4. Conclusion

MOON Lab facilitates programming of the control and data acquisition applications and gives wide capabilities and flexibility of graphical data presentation. MOON Lab is a good choice for solving of many software problems on a uniform basic and linking them together.

5. References

[1] S. A. Kryukov, 'New Programming Technologies for Beam Diagnostics', Proc. EPAC'96, CERN, Vol. 3, p. 1781 (1996)

[2] S. A. Kryukov, 'Shall we Replace LabVIEW® With Object-Oriented Instrument?', Proc. Int. Workshop PCaPAC'96, DESY, Hamburg (to be printed)

® Microsoft and Windows NT are registered trademark of Microsoft Corporation
™ Windows and Windows 95 are trademarks of Microsoft Corporation