СООБЩЕНИЯ
ОБЪЕДИНЕННОГО
ИНСТИТУТА
ЯДЕРНЫХ
ИССЛЕДОВАНИЙ

Дубна

E11-99-231

A.M.Raportirenko

LISP-BASED SYSTEM FOR THE LOGIC ANALYSIS AND VISUALIZATION OF THE DATA
(Description of the System Dynamics)

1999

Рапортиренко А.М. E11-99-231
LISP-система для логического анализа и визуализации данных
(Описание динамики системы)

Представлена мобильная система UNIX для логического анализа и визуализации данных, работающая в среде X11/Motif. Логически система состоит из трех компонентов — графического редактора, интерпретатора языка Standard LISP и модуля интерфейса между графическим редактором и LISP-интерпретатором. С помощью графического редактора, используя базовый набор графических объектов, таких как линия, прямоугольник, эллипс, текст и т.д., можно создавать более сложные составные объекты — шкалы, гистограммы, компоненты логических схем, и сохранять их в файлах как LISP-программы. Это позволяет создавать библиотеки из часто используемых сложных графических компонентов. Динамические связи между графическими атрибутами объектов реализуются через LISP-программы. Анализ отображаемых данных может осуществляться как аналитически, с использованием LISP-программы, так и численно, с использованием программ C или FORTRAN. Компилированные LISP, C и FORTRAN программы могут загружаться в память компьютера динамически во время работы системы.

Работа выполнена в Лаборатории вычислительной техники и автоматизации ОИЯИ.

Сообщение Объединенного института ядерных исследований. Дубна, 1999

Raportirenko A.M. E11-99-231
LISP-Based System for the Logic Analysis and Visualization of the Data
(Description of the System Dynamics)

The portable UNIX system for the logic analysis and visualization of the data working in X11/Motif environment is presented. The system consists of three logical components — the graphics editor, Standard LISP interpreter and the interface between the graphics editor and LISP interpreter module. With the help of the graphics editor and using a base set of graphics objects — lines, rectangles, ellipses, texts, etc. it is possible to create more complex compound objects — scales, histograms, logic circuits components, and to store them as LISP programs. It allows one to create libraries from a frequently used complex graphics components. The dynamic links between the graphics attributes of objects are realized via LISP programs. The analysis of the displayed data can be carried out as algebraically, using LISP programs, as numerically using C or FORTRAN programs. Compiled LISP, C and FORTRAN programs can be loaded into computer memory dynamically at run time.

The investigation has been performed at the Laboratory of Computing Techniques and Automation, JINR.

Communication of the Joint Institute for Nuclear Research. Dubna, 1999

1. Introduction

The portable UNIX system for the logic analysis and visualization of the data working in X11/Motif environment is presented. The system consists of three logical components—the graphics editor, Standard LISP interpreter [3] and the interface between the graphics editor and LISP interpreter module. With the help of the graphics editor and using a base set of graphics objects—lines, rectangles, ellipses, texts, ...(fig. 1.) it is possible to create more complex compound objects—scales, histograms, logic circuits components, ...(fig. 2.) and to store them as LISP programs. It allows to create libraries from a frequently used complex graphics components. The dynamic links between the graphics attributes of objects are realized via LISP programs. The analysis of the displayed data can be carried out as algebraically, using LISP programs, as numerically using C or FORTRAN programs [4]. Compiled LISP, C and FORTRAN programs can be loaded into computer memory dynamically at run time.

2. Description of the system dynamics

The interaction between the system and external data sources is accomplished via the input data objects $\mathcal{D}^{(p)}$. Each of these objects is defined with some set of attributes $d_i^{(p)}$. On object creations each of its attributes is assigned some default value $d_i^{(p)} \longleftarrow \delta_i^{(p)}$. During the system work the acquired data update the attributes' values.

Each screen graphics image corresponds to some graphics object $\mathcal{G}^{(q)}$. The graphics objects $\mathcal{G}^{(q)}$ are defined by some dynamic $g_i^{(q)}$ and static $s_j^{(q)}$ attributes and by the information on available shapes, defined by shape objects $\mathcal{S}^{(s)}$, for their representation. Like the previous case, on graphics object creation its attributes are assigned some default values $g_i^{(q)} \longleftarrow \gamma_i^{(q)}$, $s_j^{(q)} \longleftarrow \sigma_j^{(q)}$.

Depending on attributes' values, one of shape objects $\mathcal{S}^{(s)}$ for representation of graphics object on the screen is created.

The difference between dynamic and static attributes is that the values of the first depend on input data objects attributes values and graphics objects attributes

values, and the values of the latter may be changed by a user with the help of a mouse or keyboard.

As from one side, the areas of allowed values of input data objects attributes and graphics objects dynamic attributes differ significantly. In the first case it might be arbitrary LISP expressions, and in the second—only integers, strings or functional definitions (lambda expression or addresses of compiled functions). And from the other side—there is no "one to one" correspondence between input data objects attributes and dynamic attributes of graphics objects. Thus dynamic links objects $\mathcal{L}^{(r)}$ are used to set dynamic links between attributes.

Each of such objects is defined: by some set of attributes $l_i^{(r)}$; by assigning a correspondent graphics object $\mathcal{G}^{(q)}$ and input data object $\mathcal{D}^{(p)}$; by assigning links between the attributes $g_i^{(q)}$ and $l_i^{(r)}$; and dependencies $\phi_i^{(r)}(d_j^{(p)}, \ldots, l_k^{(q)}, \ldots) \longrightarrow l_i^{(r)}$. On dynamic links object creation each of its attributes is defaulted $l_i^{(r)} \longleftarrow \lambda_i^{(r)}$ or $\phi_i^{(r)}(\delta_j^{(p)}, \ldots, \lambda_k^{(q)}, \ldots)$.

As well as the correspondent graphics object $\mathcal{G}^{(q)}$ is created. The default values of attributes of dynamic links object are used to set the default attributes values for all of the objects being created. The dependencies $\phi_i^{(r)}(d_j^{(p)}, \ldots, l_k^{(q)}, \ldots)$ can be implemented as LISP, C or FORTRAN programs [1–3].

Shape objects $\mathcal{S}^{(s)}$ are defined with some attributes $s_i^{(r)}$ and by setting its content components—primitive shape objects $\mathcal{S}_\alpha^{(s)}$.

And at last primitive shape objects $\mathcal{S}_\alpha^{(s)}$ realize the base set graphics primitives (see fig. 1.) which are used to construct compound graphics objects with dynamic parameters (see fig. 2).

3. Primitive shape objects

Presently the system has 14 types of primitive shape objects—a rectangle, rounded rectangle, square, rounded square, polyline, polygon, regular polyline, regular polygon, scale, circle, ellipse, arch, elliptical arch and text. If necessary this list can be extended.

For texts displaying Type1 fonts and LISP interpreter built-in Type1 fonts rasteriser from X11 are used.

Each of objects is characterized by some set of X11 graphic attributes and geometry attributes (x, y, w, h, ϕ), where x, y–are the coordinates of the bounding box rectangle top left corner, ϕ–is the angle between the x axis and rectangle.

Height h and width w of a rectangle may have negative values. In which case the drawing direction is changed, resulting to its flipping in shifted by (x, y) and rotated by angle ϕ coordinates system.

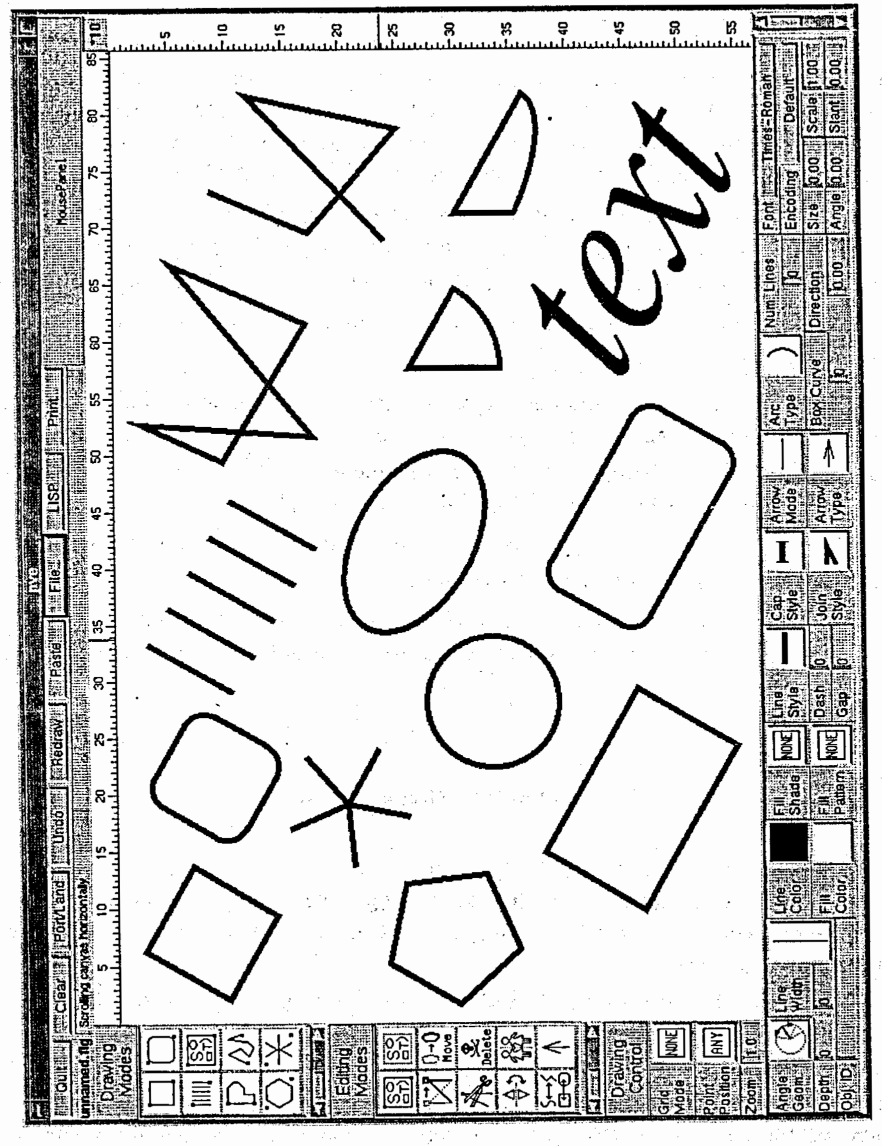
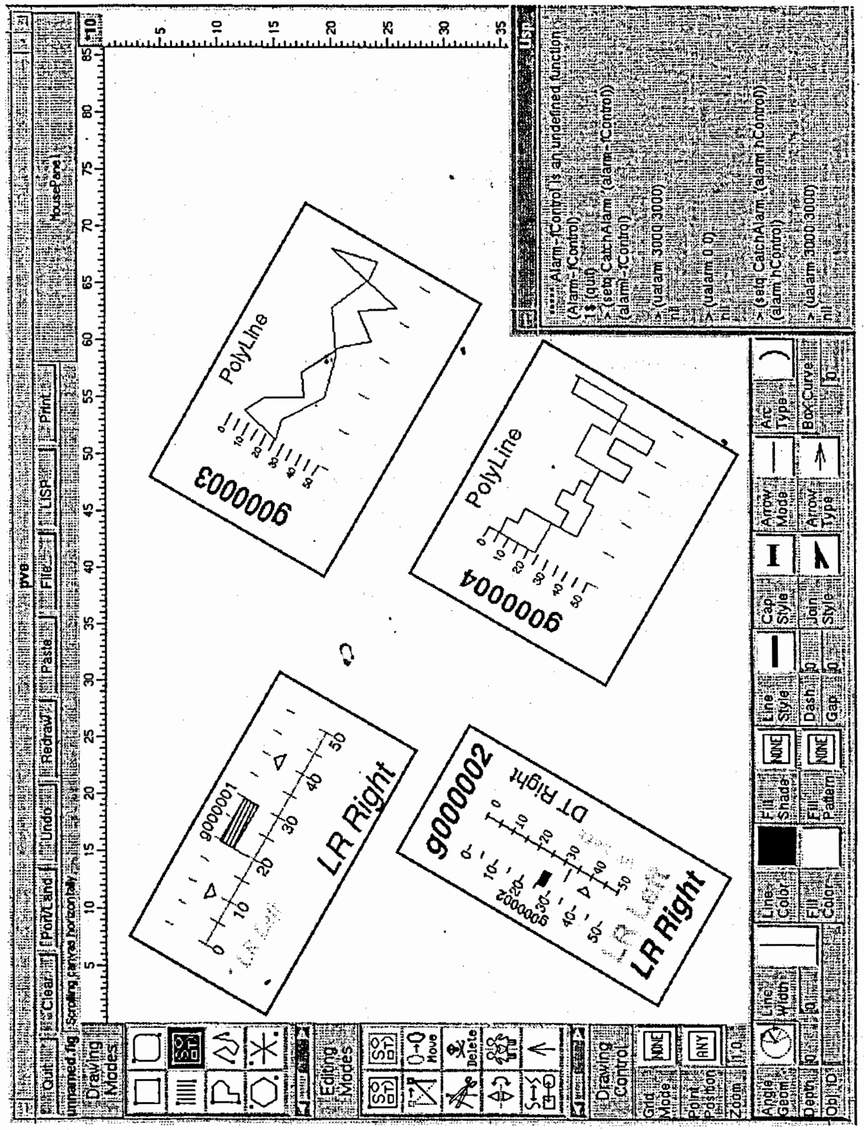


Рис. 1. Graphics primitives of system



```
***** Alarm-fControl is an undefined function
(Alarm-fControl)
1$ (quit)
> (setq CatchAlarm '(alarm-fControl))
(alarm-fControl)
> (ualarm 3000 3000)
nil
> (ualarm 0 0)
nil
> (setq CatchAlarm '(alarm hControl))
(alarm hControl)
> (ualarm 3000 3000)
nil
```

Рис. 2. Example of graphic objects of system

4. Conclusion

This system is a by-product of the author work on development of the GUI for the computer algebra system REDUCE [2]. In principle, some additional efforts being spent, it can be used in expert systems. This work was supported in part by the INTAS grant No. INTAS-96-0842.

ЛИТЕРАТУРА

1. J. B. Marti, A. C. Hearn, M. L. Griss, C. Griss,
   The Standard Lisp Report, SIGPLAN Notices, ACM, 14, (10), 48-68, (1979).

2. A. C. Hearn,
   REDUCE User's Manual, Version 3.6, RAND, Santa Monica, 1995.

3. A. M. Raportirenko,
   GSL: A portable Standard LISP interpreter, In Proceedings of the Third International Workshop on Software Engineering, Artificial Intelligence and Expert Systems for High Energy and Nuclear Physics, 1993.

4. A. M. Raportirenko,
   Использование GSL в символьных и численных вычислениях, ОИЯИ P11-99-230, Дубна, 1999.