Available online at www.sciencedirect.com

SCIENCE @ DIRECT®

Journal of Visual Languages and Computing
16 (2005) 231–244

Journal of
Visual Languages
& Computing

www.elsevier.com/locate/jvlc

A universal fast graphical user interface building tool for arbitrary interpreters

L. Pere[a,b], M. Koniorczyk[c,*,1]

[a] *Computing Services Centre and Department of Informatics and General Technology, Faculty of Science, University of Pécs, Pécs, H-7624 Pécs, Ifjúság útja 6. Hungary*

[b] *Computer and Automation Research Institute of the Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary*

[c] *Research Centre for Quantum Information, Slovak Academy of Sciences, Dubravská Cesta 9. 845 11 Bratislava, Slovakia*

Received 12 January 2004; received in revised form 13 September 2004; accepted 20 September 2004

Abstract

We consider the issue of implementing graphical user interfaces (GUIs): we present an easy-to-use and fast GUI building tool, specially designed to be used with interpreters. It supports a variety of communication methods and interaction models, therefore being able to collaborate with a huge diversity of interpreters in a natural way, in POSIX compliant (or similar) environment. Thus it enables the programmer to easily create a GUI, no matter what kind of language or model the actual interpreter implements. Event-driven programs in UNIX shells and graphical user interfaces in a data oriented language are presented as example applications.
© 2004 Elsevier Ltd. All rights reserved.

Keywords: User interface; GUI; Interpreters; Shells; Event-driven programming

*Corresponding author. Tel.: +42 190 261 0470; fax: +42 125 477 6085.
E-mail address: kmatyas@szfki.hu (M. Koniorczyk).
[1] On leave from Research Institute for Solid State Physics and Optics, Hungarian Academy of Sciences, Budapest.

1045-926X/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jvlc.2004.09.002

1. Introduction

Many of the formal languages can be implemented on a POSIX [1] compliant or at least similar system by creating an interpreter software. A large variety of problems—from system administration to database management—can easily be solved with the aid of interpreters, such as shell programs like the Bourne Again Shell (BASH), AWK, Perl, etc. When running an interpreter, a variety of command-line based external utilities can be invoked, including e.g. self-made programs, special utilities for graphics or device control, or SQL clients. Though it is sometimes claimed that the use of a command-line is out-of-date and should be replaced by entirely visual approaches, it has certain advantages in many cases. It would be desirable to extend the possible choices for a development method and its implementation where visual programming and GUI interfaces can naturally co-exist with the "old fashioned" command-line interpreters or scripting languages, extending the capabilities of each other.

The tool to be described here is of this kind: it enables the developer to easily create GUIs based on his current expertise and knowledge in a certain interpreted language.

While in the case of compiled languages, there is a fine selection of programming libraries (such as forms, Qt, GTK+, etc.) to help programmers in creating GUIs, there are rather few of such tools for interpreters.

Shell scripts and command line utilities, for instance, constitute the powerful armament of a typical UNIX user. Due to the standard and flexible nature of this user attitude, it is applicable on many (not natively UNIX) platforms. There have been endeavors to create graphical user interfaces for command line utilities or shell scripts. This progress dates back to 1991: Hesketh [2] described a button system, Perly, realizing to some extent the graphical means of running a Perl or shell script. His main concern was to provide on-screen graphical buttons connected to UNIX command scripts. When a button was pressed, the associated command script was executed. Teuben [3] suggested a method where the GUI was described in the comment lines of the shell script. Sorzano et al. [4] implemented a way of creating graphical user interfaces to UNIX commands.

An example of a more universal utility, providing dialog boxes especially for shell scripts is the `dialog` utility [5] and its affiliates: `kdialog` and `gdialog`. A typical feature of these is that the main program is written in the actual shell, while the utilities creating the GUI are called as external programs. They receive the description of the desired interface from command-line switches and write user input to their standard output. These are suitable for simple intercourses (e.g. a single question), but their application may become laborious in more complex situations. They lack the opportunity of defining actions for the widgets, thus the only interaction model feasible in this framework is a sequence of questions and answers. As a consequence, they are unsuitable for developing more sophisticated action-driven programs, which is in itself against the nature of shells. They have, however, a remarkable advantage: the method of the GUI design is independent of the shell actually used. The same syntax is used for BASH, C shell, etc. Therefore there is no need to learn a new GUI description language for each shell.