The Icon Newsletter

No. 52 – April 1, 1997

Contents

Mail-Order Program Material

Over the years, we've seen the impact of changing technologies on our distribution of Icon program material. At one time, all Icon program material was distributed on 1/2" magnetic tape and documents were provided in printed form. Then diskettes came along and in a big way. Data cartridges emerged but were never in much demand. As diskette technology changed, we went from low-density, one-sided 5.25" floppies to high-density 3.5" floppies.

The demand for magnetic tape disappeared long ago except for an occasional request for an implementation that wasn't available on diskette. In the last 12 months, we've only had only one request for a magnetic tape.

The biggest impact came with the Internet and increasingly easy access to on-line files. For years now, FTP has been the major method for the distribution of Icon program material and documentation. The number of downloads from our FTP area varies considerably with the season, the academic calendar, and releases of new material. It averages about 4,500 files a month — nearly 55,000 files a year.

Magnetic tape isn't the only medium that's been swept away by change. We now get very few requests for diskettes — only five in the last year.

Maintaining mail-order distribution of Icon material is a considerable effort, even when the demand is low. For example, we need to make up new master diskettes and make copies for stock whenever there is a new version of Icon — on a per-platform basis. A new version of MS-DOS Icon, for example, means new masters and new stock.

That's been only "in theory" for a year or so. We haven't made up new masters, much less copies, until we had a request. Version 9.3 of Icon was released last fall; we have yet to have a request for it on diskettes.

For these reasons, we are discontinuing our mail-order distribution of Icon program material. If there's something you just have to have on diskette and we happen to have a copy around, we'll sell it to you. But we no longer maintain and upgrade our stock.

Magnetic tapes fall into the nostalgia classification for us (following punched cards and paper tape that went that way long ago). Diskettes aren't there yet, and they probably will be around for some time for our distribution of updates to the source code for Icon and the Icon program library.

Icon on the Web

Icon is on the World Wide Web at

http://www.cs.arizona.edu/icon/

Teaching Icon

In the last issue of the *Newsletter* Bill Mitchell, a former student here, wrote about teaching Icon in our comparative programming language course.

Another former student, Beth Weiss, taught the course before Bill. She's no longer here, but she left us some notes on her views about teaching Icon and an outline from the Icon part of the course.

Here's what she had to say:

> Some students really "get it", and others can't understand why they couldn't just do everything in C.
>
> I find it very helpful to repeatedly ask the question "And what would it take to do this in C or Pascal?" It keeps them thinking about the dynamic nature of Icon, which many of them have come to like a great deal. I push string scanning :–), and most students eventually do come to realize that scanning is better than array manipulation, and they appreciate its power. Generation seems to be something they finally come to understand in Icon once they see it in Prolog later in the semester.
>
> I try very hard to stay away from the "see how compactly this can be written in Icon one-liners", since I think that type of code is counter-productive in terms of thinking and coding ability. Some people, of course, live for that type of code, and there's at least one every semester. :–)
>
> One other thing that I find necessary in the course is to keep reminding them why Icon is a useful tool for them to learn. I can't deny it's my favorite language :–), but that's not the reason we teach it in comparative programming languages, of course.

Downloading Icon Material

Most implementations of Icon are available for downloading via anonymous FTP:

ftp.cs.arizona.edu (cd /icon)

Beth used 74 transparencies for the Icon portion of her course. Here's the outline as taken from the transparencies, omitting examples and only showing subtopics in a few places:

- Introduction
- Characteristics of Icon
  - expression-based
  - goal-directed evaluation
  - no variable declarations
  - high-level string scanning
  - dynamic structures
  - automatic storage management
- Success and failure
- Conditional expressions
- Bounded expressions
- Output
- Procedures
- Outcome of an expression
- Goal-directed evaluation
- Control backtracking
- Iteration
- Alternation
- Conjunction
- Loops
- Selection (case expressions)
- Augmented assignment
- Variables, values, and results
- String scanning
  - the concept
  - positions in strings
  - matching functions
  - substrings
- Alternation and scanning
- Concatenation
- Csets
- Analysis with csets
- Procedures and generators
- Other string functions
- Character generation

Substrings

Lexical comparison

Structures

Lists

Tables

In a course like this, where several programming languages are taught in one semester, the time for any one language is limited. Bill Mitchell omitted co-expressions and records because of this. Note that Beth omitted these as well as sets.

The Icon Newsletter

Ralph E. Griswold, Madge T. Griswold,
and Gregg M. Townsend

Editors

The Icon Newsletter is published three times a year and is available on the World Wide Web. To receive printed copies, contact:

Icon Project
Department of Computer Science
The University of Arizona
P.O. Box 210077
Tucson, Arizona 85721-0077
U.S.A.

voice: (520) 621-6613

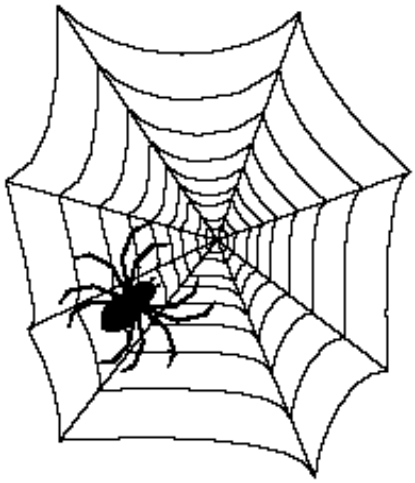
fax: (520) 621-4246

e-mail: icon–project@cs.arizona.edu

THE UNIVERSITY OF
ARIZONA
TUCSON ARIZONA ®

and

Bright Forest Publishers
Tucson Arizona



Web Links

The Icon Web Site

The Icon Web site is updated when things related to Icon happen, such as a new Icon program library release, a new issue of this *Newsletter,* and so on.

If you're actively interested in Icon, you should follow the Status link on the Icon home page for information about new developments.

We also add new technical reports (Icon Project documents, or IPDs) from time to time, as well as a few older ones that still are useful but have not been prepared as Web pages before. You can find these by following the Technical Reports link in the **Documentation** section of the Icon home page. Check the Other Documents link too; it leads to documents on Icon that are not in the IPD format.

Other Pages

Marc Espie's page on Icon has an interesting game called X-Tiles that is written in Icon. There's also a paper on Chinese monoids in which the combinatorial computations were done using Icon. Check this out at

http://www.eleves.ens.fr:8080/home/espie/icon/

Tom Christopher's home page has links to several worthwhile Icon programs and related material. It's at

http://www.iit.edu/~tc/

If you have a Web site with interesting material related to Icon and would like us to list it in the *Newsletter,* let us know.

Native Interface Components in Windows Icon 9.3

Editors' Note: The following article was contributed by Clint Jeffery, who is implementing Version 9 of Icon for Windows.

Icon's graphics facilities use a combination of built-in functions and Icon program library (IPL) procedures to perform operations such as opening and closing windows, drawing on them, and reading events. Icon's user interface components (menus, scroll bars, dialogs, etc.) are all built using IPL procedures written in Icon. This allows them to run on the largest possible number of machines, since they do not depend on proprietary component sets such as Motif. Icon's IPL procedures happen to implement a look and feel similar to Motif, which is attractive and feels natural on most UNIX platforms.

Enter Windows, or more specifically Windows Icon. In versions prior to 9.3, released in various beta tests, the "Motif" look of the user interface components proved to be unpopular with Windows users. For this reason, Version 9.3 of Windows Icon introduces various built-in user interface components that make use of native Windows features

Description of the Native Facilities

Version 9.3 of Windows Icon provides the following native facilities. Additions are still being made to this set. The facilities described here are available now, but they are still being integrated into the Icon program library so that they are used automatically in Icon programs that provide a visual interface. This section summarizes features; they are described more fully in the Windows Icon documentation, IPD271, which comes with the distribution.

The goal of the native facilities is not to provide the entire Windows repertoire, any more than the entire X Window repertoire is provided to UNIX users. Instead, features have been chosen that are (1) important to the Windows look and feel, (2) general enough to be implementable on other platforms, and (3) can coexist or be integrated with existing IPL facilities.

Menus

A menu bar is created with a call like:

```
MenuBar(W,
 ["&File", "&Open", "&Save", "&Exit"],
 ["&Edit", "&Cut", "&Paste", "&Copy"],
 ["&Help", "&About"]
 )
```

This function converts approximately the top text line of the window into a menu bar. The appearance of the above example is given in Figure 1. When menu items are selected, they are produced as entire strings (such as "&Open") by Event().

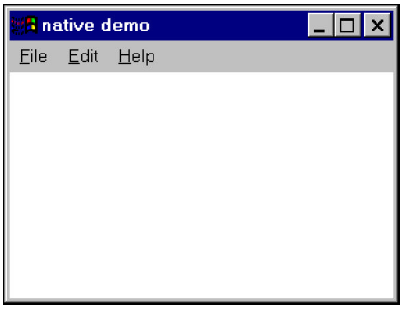


Figure 1: A Windows Menu Bar

Scroll Bars

A scroll bar is created with a call like

```
ScrollBar(W, "sb_1", x, y, wd, ht)
```

This function places a scroll bar with a particular size and position, which default to a standard size on the right edge of the window. The appearance of a typical scroll bar is illustrated in Figure 2. When scroll bar activity takes place, the scroll

bar's string id is produced (in this case, "sb_1") by Event(), and &x and &y are both set to the scroll bar's position.

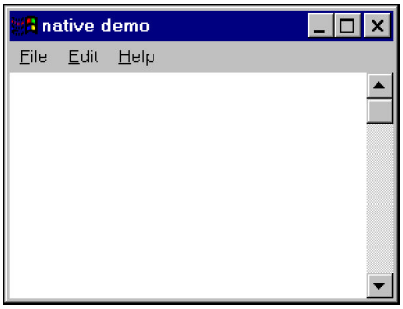


Figure 2: A Windows Scroll Bar

Buttons

A button is created with a call like

```
Button(W, "hello", x, y, wd, ht)
```

This function places a button with a particular size and position. The size defaults to a standard size large enough display the button's label. The appearance of a pair of buttons is illustrated in Figure 3. When a button is pressed, the button's string label is produced (in this case, "hello") by Event().

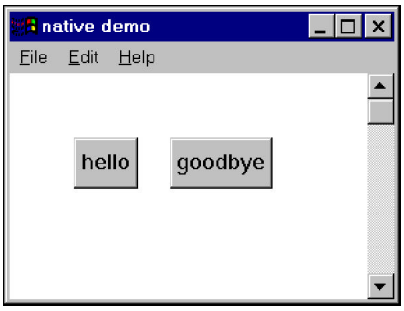


Figure 3: A Pair of Windows Buttons

Common Dialogs

Several common dialogs are provided for selecting colors, fonts, and files to open or save. These functions return an attribute value or a file name. Examples are illustrated in Figures 4 - 7.

Conclusion

For developing elaborate Windows interfaces, Icon does not compete with commercial Windows-specific programming environments such as Visual Basic or Delphi. Still, Icon programmers can write portable

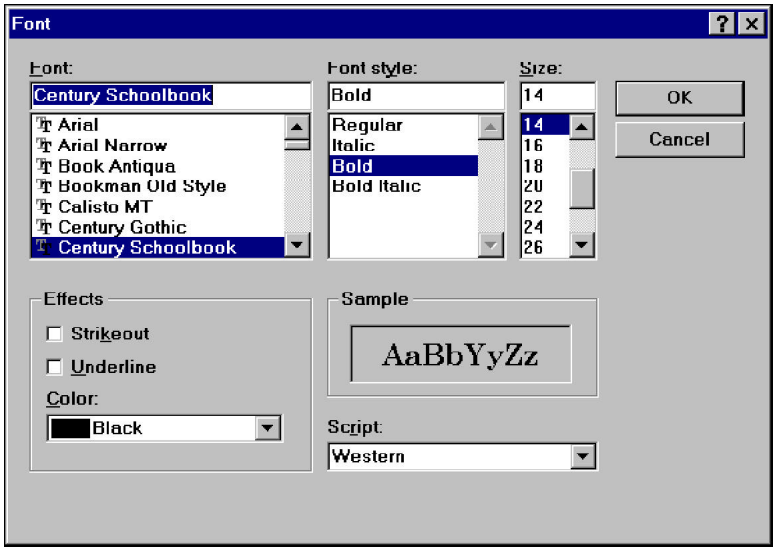


Figure 4: The Windows Font Dialog

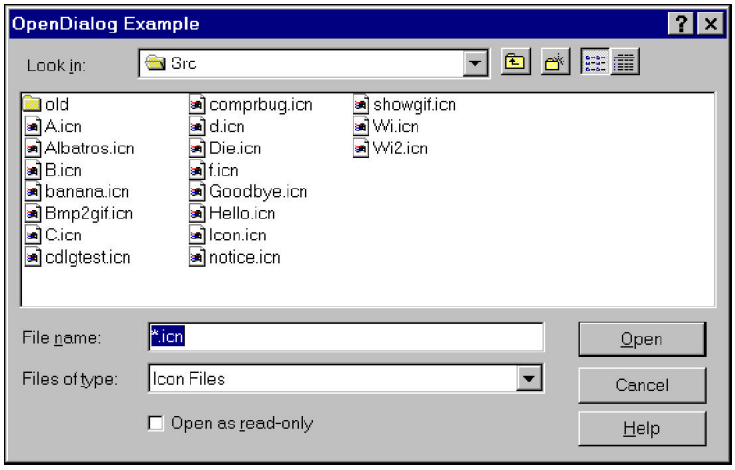


Figure 5: The Windows Open Dialog

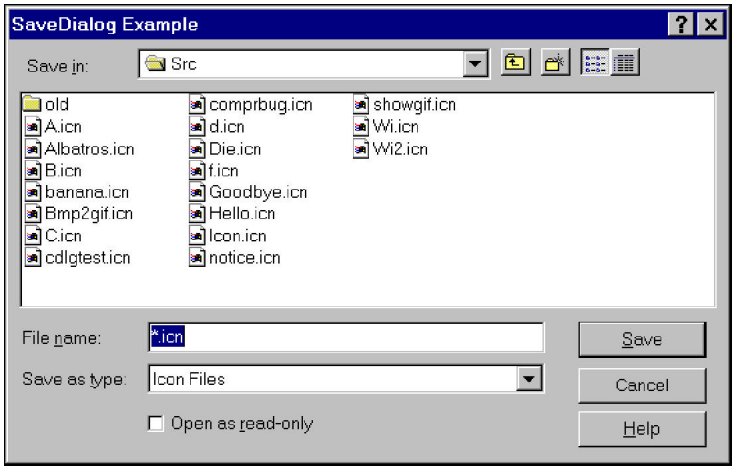


Figure 6: The Windows Save Dialog

applications that take advantage of Icon's power on multiple platforms. The native facilities make it easy to construct applications that look and feel like they belong on the Windows platforms. When integration with the IPL is complete, this look and feel will come to existing Icon programs written for the X Window System with no source modification. Some additional native facilities, such as edit regions and the multimedia interface, are not described in this article. Additional information can be requested and comments sent by e-mail to Clint Jeffery:

jeffery@cs.utsa.edu

Winows Icon is available from:

ftp://ringer.cs.utsa.edu/pub/icon/nt/graphics

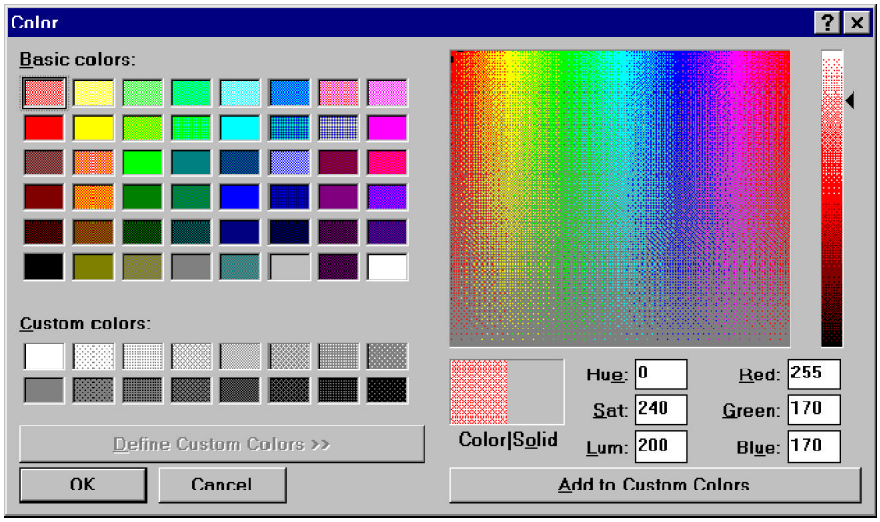


Figure 7: The Windows Color Dialog

Programming Language Handbook

Macmillan is planning a multi-volume **Handbook of Programming Languages**. Ralph Griswold has contracted to write a section on Icon for **Volume II: Imperative Languages**. The section on Icon is projected to run about 100 pages.

We don't know yet when the handbook will appear, but our contribution is due August 1, 1997. We'll provide more information when we have it.

From Our Mail

What is Icon used for?

Answering that question is not as hard as knowing how many Icon programmers there are. Of course, we can't know everything Icon is used for, but we know it is used for rapid prototyping of complex applications, research in the humanities, file processing, systems programming, molecular biology, and so on. One time we did an "A to Z" listing of specific applications and found most letters had at least one entry. Basically, the applications of Icon are very diverse. After all, Icon is a high-level, general-purpose programming language. Despite its emphasis on facilities for processing text and complex structures, it can be used for most anything. Interestingly, Icon seems to be used most either for very simple applications or very complex ones.

I have a new boss who says I have to program in C++, but there are plenty of things that I can do faster and more easily in Icon. How can I talk my boss into letting me use Icon for these tasks?

We don't encourage you to do anything that might get you into trouble, although we know several persons who have done very well using Icon without telling anyone. If the programs you have in mind aren't going to be used by anyone else and there is no issue with respect to future maintenance, whatever programming language is the best for getting the job done ideally should be used. "You must use X" often is just a simple rule that makes life easy for a supervisor and removes a possible source of criticism. But to address your specific question, one way to convince your supervisor is to demonstrate that the use of Icon can save time and money without putting anything at risk. After all, most organizations welcome increased productivity.

There used to be a technical report on the interface vidgets, but I can't find it anymore. Where is it?

The vidgets have changed substantially since the original description was written — so much so that the old document can no longer be used. We don't have the resources to update the document; it's just something that has passed from the scene. The vidgets are available through VIB and various dialogs.

Knowledge Explorer

Editors' Note: Dick McCullough has written an Icon tool for organizing knowledge. His description follows.

The Knowledge Explorer (ke) is an interactive tool for organizing knowledge. It can be viewed as a super-intelligent filing system, which will restructure itself on command. The foundation of ke is my knowledge representation language (kr), which is based on a unique theory of knowledge developed by Ayn Rand (*Introduction to Objectivist Epistemology*, Expanded Second Edition, Meridian, 1990).

kr is a subset of English, with keywords that have a specialized meanings. Sentences that ke does not understand are simply recorded as "newstatement"s. kr uses lists, sets, and associative tables to represent knowledge, which is why I chose to implement it using Icon (Ralph E. Griswold and Madge T. Griswold, *The Icon Programming Language*, Third Edition, Peer-to-Peer Commmunications, 1996). The pattern matching capabilities of Icon were used to implement a front end parser for kr. During the development of the initial version of ke, from October 1996 to March 1997, I made significant changes every day, and did major rewrites about once a month. This rapid evolution would not have been possible without using Icon.

A small sampling of statements in kr is shown in the box below.

I have made the Knowledge Explorer available on the Internet as a shareware product. It can be downloaded from

ftp://ftp.cdepot.net/ke/

Binary versions are available for Windows and UNIX. For details, contact:

Dick McCullough
rhm@cdepot.net

```
animal  ise  man,cat,dog                                  # hierarchy
Dick   isa person                                         # hierarchy
Dick   is  "Richard H. McCullough"                        # identity/alias
Dick   has sex:male                                       # characterization
Dick   do  go to store                                    # event – unit of change
man     is  animal with rational                          # definition
phone–list  isa  relation                                 # define  &  load  relation
phone–list  has  r_role:  "phone:1    person:2"
phone–list  has  r_meaning:  "person:2  has  phone:phone:1"
read  phone–list.rel
at space,time,view \                                      # define  new  knowledge  unit
          from here,now,tabula_rasa
ke      has  ?                                            # display all characteristics
apple   ?    orange                                       # determine  relation  of  concepts
apple   isa   ?                                           # walk up hierarchy
animal  ise   ?                                           # display subhierarchy
check     definition                                      # display  undefined concepts
hfocus  :=  event,newword,newstatement                    #  hierarchy  printout
hformat  :=  outline                                      #  hierarchy  printout
```