VECTOR

APL96 Call for Participation

Win a Free Registration - see page 107

The Journal of the British APL Association

ISSN 0955-1433

Vol.12 No.3 January 1996

A Specialist Group of the British Computer Society

Contributions

All contributions to VECTOR may be sent to the Journal Editor at the address on the inside back cover. Letters and articles are welcome on any topic of interest to the APL community. These do not need to be limited to APL themes, nor must they be supportive of the language. Articles should be accompanied by as much visual material as possible (b/w or colour prints welcome). Unless otherwise specified, each item will be considered for publication as a personal statement by the author. The Editor accepts no responsibility for the contents of sustaining members' news, or advertising.

Please supply as much material as possible in machine-readable form, ideally as a simple ASCII text file on an IBM PC compatible diskette (any format). APL code can be accepted as camera-ready copy, in workspaces from I-APL, APL*PLUS, IBM APL2/PC or Dyalog APL/W, or in documents from Windows Write (use the Vector TrueType font, available free from Vector Production), and Winword-2.

Except where indicated, items in VECTOR may be freely reprinted with appropriate acknowledgement. Please inform the Editor of your intention to re-use material from VECTOR.

Membership Rates 1995-96

Category	Fee	Vectors	Passes
UK Private	£12	1	1
Overseas Private	£14	1	1
(Supplement for Airmail, not needed for Europe)	£4		
UK Corporate Membership	£100	10	5
Overseas Corporate	£135	10	
Sustaining	£430	10	5
Non-voting Member (Student, OAP, unemployed)	£6	1	1

The membership year normally runs from 1st May to 30th April. Applications for membership should be made to the Administrator using the form on the inside back page of VECTOR. Passes are required for entry to some association events, and for voting at the Annual General Meeting. Applications for student membership will be accepted on a recommendation from the course supervisor. Overseas membership rates cover VECTOR surface mail, and may be paid in sterling, or by Visa, Mastercard or JCB, at the prevailing exchange rate.

Corporate membership is offered to organisations where APL is in professional use. Corporate members receive 10 copies of VECTOR, and are offered group attendance at association meetings. A contact person must be identified for all communications.

Sustaining membership is offered to companies trading in APL products; this is seen as a method of promoting the growth of APL interest and activity. As well as receiving public acknowledgement for their sponsorship, sustaining members receive bulk copies of VECTOR, and are offered news listings in each issue.

Advertising

Advertisements in VECTOR should be submitted in typeset camera-ready format (A4 or A5) with a 20mm blank border after reduction. Illustrations should be photographs (b/w or colour prints) or line drawings. Rates (excl VAT) are £250 per full page, £125 for half-page or less (there is a £75 surcharge per page if spot colour is required).

Deadlines for bookings and copy are given under the Quick Reference Diary. Advertisements should be booked with, and sent to: Gill Smith, Brook House, Gilling East, YORK YO6 4JJ. Tel: 01439-788385 CompuServe: 100331,644

Contents

Great Things Come in Three's...

Announcing: **J** Release 3

Now Available via OLE...

J Release 3 for Windows 95/NT from Iverson Software makes access to J a breeze for users of Visual Basic, Delphi or any OLE-compliant software. Select the J EXE local server for development with full access to the J system, or the J DLL in-process server for optimal run-time performance. Both provide an incredibly powerful programming language, whose clean design and lack of special requirements makes a smooth fit with other Windows software.

Prefer to develop you applications only in J? You get a sophisticated Windows integrated development environment, with built-in script and forms editors, a session manager with customizable menus, support for multiple script and execution sessions, a simple yet effective Windows interface, access to virtually any database, and much more.

OLE support is provided with the Professional Edition, which also includes royalty-free runtime use. J Release 3 is also available in Standard and Educational Editions, and for other systems, including PC DOS, Windows 3.1, Linux, Mac , Sun and RS/6000.

For more information, contact Anne Faust,Strand Software, 19235 Covington Court, Shorewood, MN 55331, Tel (612) 470-7345, Fax (612) 470-9202, email: amfaust@aol.com. Also available through dealers worldwide.

Editorial: the Place of APL

APL both is computing and is not computing. *APL* is also a British Limited Company. It is computing because it is in widespread commercial use, because without this use it would probably never have been developed and enhanced, and because without the mechanism to execute it, an executable imperative notation would be hobbled.

It is not computing because it was invented before it was thought that a computer might execute it, because it is a notation first and a programming language second and because its ethos and its practitioners don't fit into the standard computing framework.

Your editor has, for years, advocated that the British APL Association should become an independent associate of the British Computer Society instead of a wholly owned subsidiary. Your editor has not carried the British APL Association with him.

In 1986 the Association agreed that your editor (then Hon Sec of the Association) should form a company called *APL Projects Limited* (*APL* for short!) which it was then hoped could be commissioned to do work for the Association. Not much came of it. The views of the Committee changed and *APL* was never commissioned to do anything. It is a dormant company and has never traded. It could run conferences.

Several prominent APLers liked the suggestion made in *Vector* that an independent international organisation be set up to manage APL conferences. While ACM was considering the APL96 TMRF, members of the SigAPL board were ready (as Plan B) to ask APL Projects Limited to run it. Can *APL* become Plan A for APL97 or APL98?

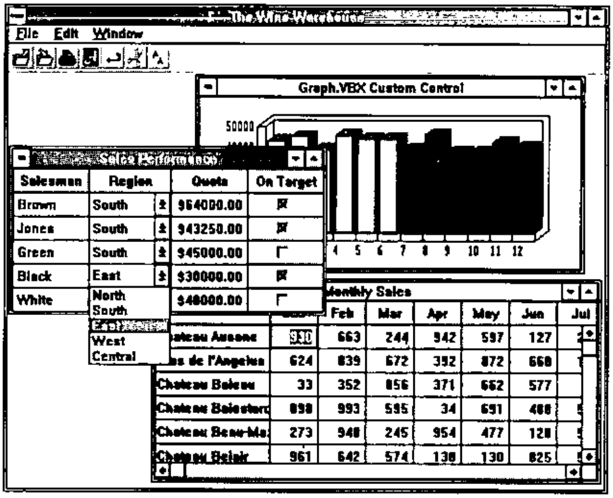
The suggestion is that *APL* should sell one share to each APL group that wants to join and will appoint a Director from each of the groups — a nominated trusted APLer. The groups will then lend *APL* as much as each can afford, to sponsor it to run APLnn.

APL would aim to run profitable conferences and to repay the loans. If it pays dividends these will of course be an equal amount for each share but most of the profit would be retained so that the following year there would be some seed finance and eventually borrowing will become unnecessary.

If your group wants to join in, make comments or suggestions about it, please contact me (address inside back cover). It isn't too early to start on APL98!

dyalog APL

The Definitive APL for Windows™



Salesman	Region	Quota	On Target
Brown	South	$64000.00	✓
Jones	South	$43250.00	✓
Green	South	$45000.00	☐
Black	East	$30000.00	✓
White		$40000.00	☐

		Feb	Mar	Apr	May	Jun	Jul
ateau Ausone	930	663	244	942	597	127	[illegible]
s de l'Angelus	624	839	672	392	872	668	[illegible]
Chateau Baleau	33	352	856	371	662	577	
Chateau Balestard	898	993	595	34	691	408	[illegible]
Chateau Beau-Ma	273	948	245	954	477	128	[illegible]
Chateau Belair	961	642	574	138	130	825	[illegible]

Dyalog APL/W Version 7.1 includes a sophisticated built-in Grid object, with numeric, currency, and date fields as well as combo boxes and button cells. The Grid also provides an undo feature, cut and paste facilities (which let you move data quickly and easily between APL and your favourite spreadsheet), resizable rows, columns and titles, and drag/drop editing. Dyalog APL/W supports Visual Basic Custom Controls, ToolBar, StatusBar and TabBar objects, automatic Hints and Tips, Metafiles, MDI, 3-D Forms and Controls, a fully customizable Session, an ODBC interface, namespaces for encapsulation, and a host of other features; all designed to make it easy to develop fast, responsive and attractive Windows applications.

That is why Dyalog APL/W remains the professional choice. For further details, contact Dyadic or your local distributor today.

MICROSOFT® WINDOWS™ COMPATIBLE

Dyadic Systems Limited, Riverside View, Basing Road, Old Basing, Basingstoke, Hampshire, RG24 7AL, United Kingdom.
Tel:+44 1256 811125 Fax: +44 1256 811130 Email: sales@dyadic.com.

Quick Reference Diary 1996

Date	Venue	Event
24-25 June 1996	Toronto Canada	J User Conference
July 28 - August 2 1996	Lancaster University UK	APL96 See *Call for Papers* on page 6-9

A *J User Conference* will be held in Toronto on Monday and Tuesday, June 24-25th, 1996. The agenda will include several invited papers, tutorials and panel discussions on J, as well as presentations on current development work.

There is no attendance fee, but we do require prior registration in order to ensure we have reserved enough accommodation.

For more information, contact **Anne Faust** at:

email: amfaust@aol.com
tel: +1 612-470-7345
fax: +1 612-470-9202

Dates for Future Issues of VECTOR

	Vol.12 No.4	Vol.13 No.1	Vol.13 No.2
Copy date	1st March 96	24th May 96	6th Sept 96
Ad booking	10th March 96	31st May 96	13th Sept 96
Ad Copy	20th March 96	7th June 96	20th Sept 96
Distribution	April 96	July (at APL96)	October 96

APL96: DESIGNING THE FUTURE

July 28 to August 1, 1996
University of Lancaster, UK

Invitation and Call for Participation

We are pleased to announce the convening of the 1996 International Conference on APL, which will be sponsored jointly by ACM/SigAPL, the British APL Association, and the APL Club of Germany. The theme of the conference is *Designing the Future*. This year, the focus will be on tools, techniques, technologies, and applications that bring APL to the leading edge of computer technology. The design methods and distributed systems development techniques to be highlighted point to a bright future for APL as we anticipate the millennium.

The APL96 Program Committee seeks papers that showcase the place of array programming languages in the technological tools available for research, development, design, and delivery in any discipline. Papers focusing on APL, J, or other array programming languages are solicited. Tracks with specific industry focus will provide to participants the opportunity to explore these tools in depth with colleagues sharing their interests.

Complementing the contributed papers, APL96 will offer additional value to software developers and applications designers interested in APL for the tasks at hand. Plenary sessions will feature prominent speakers from several domains. Heinz Roggenkemper (leader of the distributed systems project at SAP, one of the world's most successful providers of enterprise-wide software systems) will speak on the use of APL in developing the Distributed Architect tool for the SAP system. A representative from Bang & Olufsen will speak on the use of constraint logic programming in the HiFi house of the future.

A focus of the conference will be a suite of workshops designed to offer participants opportunity for in-depth discussion of topics important in designing the future using APL. A full schedule of tutorials for users of APL who are new to the advanced technologies will offer additional opportunity for technical education.

Proposed Workshops, Tutorials, and Presentations

- **Nested Arrays — Get it Right First Time!** will build on work done at APL95. Norman Thomson and Phil Benkard (APL2) and Chris Burke (J) will speak.
- **Designing for Windows 95.** How to get that elusive Microsoft Logo on your software! Adrian Smith and Duncan Pearson will lead a Dyalog APL stream, supported by Dyadic Systems, and Strand Software will run a similar session on programming for Windows in J.
- **Communication among Co-operating Systems.** Insight and Soliton will run workshops on ODBC. Timo Laurmaa will show us how to do it "in the raw" with TCP/IP and shared variables under APL2. Ed Shaw will run a session on Electronic Data Interchange (EDI) which is the inter-company standard for transfer of orders and other supply-chain data.
- **Algorithms and other Tools of the Trade.** The Bang & Olufsen session will lead into a hands-on workshop. Andreas Geyer-Schulz will expand on the Genetic Algorithms work he presented at APL95. There will also be sessions on business graphics, forecasting, and algorithms used in the financial and insurance industries.

Contributed papers will be accepted on any topic relevant to the conference theme, but are particularly sought in the areas outlined above. Papers focusing on APL, J, or other array programming languages will be considered for inclusion in the conference.

Other Participation

All suggestions and contributions for panels, workshops, and tutorials are welcomed by the conference programme chairmen. A software exchange will offer a library of useful APL tools, for which contributions are also sought.

Paper Submission

Please notify either programme chairman via e-mail of your intention to submit a paper as soon as possible. Prospective authors will be given complete formatting instructions for draft and final paper submission.

Draft papers (not longer than 5000 words) are due no later than February 29, 1996. To avoid mailing delays, submission by e-mail is encouraged. Notification of the programme committee decision will be sent to authors by March 15, 1996.

Camera-ready copy of final papers must be in the hands of the proceedings editor no later than May 10, 1996. Editor **James Boyd** will assist authors with preparation of final, camera-ready copy from electronic text. Authors of papers included in the proceedings should plan to present them at the conference.

The APL96 Software Exchange:

SIGAPL invites you to send new and useful software to the APL96 Software Exchange. The Software Exchange is a good way to make your, or your firm's, skills and products more widely known. New GUI software is especially welcome. Software from prior APL conferences is also welcome if it has been significantly improved or updated.

Software received by April 30, 1996 will be available to attendees at APL96. The conference package plus later submissions will be available by about October 1996 from SIGAPL, the BBS\APL, and via ftp at watserv1.uwaterloo.ca. Small files via UUENCODE, or further info, see addresses. Software in APL, J, and related array languages is invited. English is preferred, and other languages are also welcome. Include an ASCII read.me (lisez.moi, lis-mich.dok) file that briefly describes what the software does, and what software and hardware is needed to use it.

This year we want to try something new. If possible, please also include an APLASCII (v1.4) version of any software that you submit (ftp APLASCII for your interpreter via watservel.uwaterloo.ca). You must include written permission to distribute any copyright software. Without permission, we can't distribute copyright software.

Conference Chairman:
Dieter Lattermann
Rheinstraße 23,
D-69190 Walldorf, Germany.
Tel: +49 6227-63469
Email: 100332.1461@compuserve.com

Programme and Proceedings

Co-chair: J. Philip Benkard, IBM (Retired) (USA)
jpb@acm.org

Co-chair: Adrian Smith, Causeway Graphical Systems (UK)
100331.644@compuserve.com

Proceedings Editor: James Boyd
boyd@cloud9.net

Volunteer Helpers:
Robert Bernecky, Snake Island Research, Inc. (Canada)
bernecky@eecg.toronto.edu
James A. Brown, IBM Corp.(USA)
aplbrown@vnet.ibm.com
Robert G. Brown, Lingo Allegro, Inc. (USA)
bob@acm.org
David Eastwood, MicroAPL (UK)
MicroAPL@microapl.demon.co.uk
Garth Foster, Syracuse University (USA)
gfoster@cat.syr.edu
Morten Kromberg, Insight Systems (Denmark)
insight@inet.uni-c.dk
Eric Lescasse, Uniware (France)
70731.3233@compuserve.com
John Scholes, Dyadic Systems, Ltd. (UK)
scholes@dyadic.com
Lynne C. Shaw, Consultant (USA)
shaw@acm.org
Alan Sykes, European Business Mgmt School (UK)
a.m.sykes@swansea.ac.uk

Postal addresses for abstracts, draft papers, software exchange contributions, and other offers of participation:

Papers (USA)
J.P. Benkard
21B Heritage Hills
Somers, NY 10589, USA

Papers (UK)
Adrian Smith
Brook House
Gilling East
YORK YO6 4JJ UK

Software Exchange (3.5" disk):
Dick Holt 3802
N. Richmond St.
Arlington VA 22207 USA.

newleaf

Reporting functions in APL were always tedious, but rarely difficult. A page had 66 lines and 132 columns, and all the characters were the same width. Even wrapping a paragraph of free text (like this) was a problem which could be solved in a few lines of code.

Then along came proportional fonts and users who could turn the paper around, and suddenly life got a lot harder. That is why Causeway wrote Newleaf. We wanted to print simple tables of data straight from APL without worrying about font, or page-size, or alignment. A bit like this, in fact ...

Engine	Type	Nature
Thomas	0-6-0	A cheeky little blue tank engine which started the whole thing off back in 1945
Gordon	4-6-2	Very much an express engine, and always concerned to keep up appearances. Would not be seen dead shunting his own coaches, or pulling trucks
Henry	4-6-0	Another main-line engine, who once had a bad experience with an elephant in a tunnel
Duck	0-6-0	This is Dyalog APL, after all

Here is the APL code to typeset the table, which is a simple 4 by 3 array of text vectors:

```
leaf.Grid 64 128                               ⍝ Column boundaries in points
leaf.Rowtitles'Engine' 'Type' 'Nature'         ⍝ Repeats on continuation pages
leaf.Titlefont'sib,12'                         ⍝ Stone Informal 12 point bold
leaf.Align'right' 'centre' 'left'              ⍝ Vector of column alignments
leaf.Font'si,12'                               ⍝ Font(s) for each column
leaf.Spread engines ⋄ leaf.Endgrid             ⍝ Set the text and finish off
```

Of course NewLeaf does a lot more than just tables. It handles complex page layouts (with running headers and multiple columns); it gives you precise control of frame positioning (ideal for the address box on form letters); it handles indents and bulleted lists; it allows you to include bitmaps (with wrap-around) and graphics from Rain. It has a multi-page preview tool, with built-in zoom and pan. It typeset this advertisement!

In fact if you have any printing requirements it can't do, we would like to know about them. Call us now for a fully-working preview copy, and help us to help you get the best out of your Dyalog APL environment.

Causeway Graphical Systems Ltd
5 The Maltings, Castlegate
Malton, North Yorks YO17 0DP

Tel: +44 (0) 1653 696760
Email: causeway@compuserve.com

Causeway

News from Sustaining Members

Compiled by Gill Smith

Soliton Associates

Soliton Associates is pleased to announce the availability of SQAPL for SHARP APL UNIX. SQAPL provides SHARP APL UNIX with generic client and server interfaces for the Open Systems Environment.

The SQAPL Client provides an interface from SHARP APL UNIX to standard SQL database products such as Oracle, Sybase and DB2. The SQL databases accessed by the SQAPL Client can be on the same computer as SHARP APL UNIX or on network-connected computers. The SQAPL Client allows SHARP APL UNIX to fetch data from SQL servers for use by APL, and to store APL data in SQL servers for use by non-APL applications and standard reporting tools.

The SQAPL Client can also store APL arrays as binary SQL objects with preservation of APL data structure. This allows an SQL server to be used instead of an APL file system as a repository for APL data. The data representation used by SQAPL is APL vendor independent, so that, for example, APL arrays stored by SHARP APL UNIX in Oracle as binary objects can be accessed by Dyalog APL for Windows.

The SQAPL Server provides a generic interface to SHARP APL UNIX server applications. Client applications accessing the SQAPL Server can be ODBC-compliant end-user products such Lotus 1-2-3, Microsoft Excel and Microsoft Access, or they can be custom applications implemented in standard languages such as C/C++, Visual Basic, Digital Smalltalk/V and APL. APL-coded client applications require the SQAPL Client, which is available for SHARP APL, Dyalog APL, APL*Plus and APL2.

If both components of a client/server application are APL-coded, the client application can use the SQAPL Remote Procedure Call to call the server application. The SQAPL Remote Procedure Call passes APL arrays as arguments, and returns APL arrays as results. SQAPL performs automatic conversion between different APL vendors' data representations, so that, for example, a Dyalog APL client application on a Microsoft Windows platform can call a SHARP APL server application on a UNIX platform.

The networking and interfacing capabilities of SQAPL are provided by SequeLink Data Direct, a product of Intersolv Inc. Soliton Associates distributes and supports SequeLink Data Direct for use with SHARP APL. SQAPL is a product of Insight Systems ApS. Soliton Associates distributes and supports SQAPL for SHARP APL UNIX.

SequeLink Data Direct supports a wide range of database products, hardware platforms and network protocols. The use of SequeLink reduces application development and maintenance effort, and enhances application portability and scalability.

SQL database products supported by SequeLink Data Direct include Oracle, Sybase SQL Server, Microsoft SQL Server, Informix, CA-Ingres, Rdb (for Open VMS), DB400 (for OS/400), DB2 (for AIX, OS/2 and MVS), Adabas and Teradata.

Server platforms supported by SequeLink Data Direct include OS/2, Windows NT, UNIX, Open VMS, AS/400 and IBM MVS.

Client platforms supported by SequeLink Data Direct include Windows, Windows 95, Windows NT, Open VMS, OS/2, Macintosh, UNIX and AS/400.

Networks supported by SequeLink Data Direct include TCP/IP, Novell SPX, NetBIOS, NetBEUI, AppleTalk, APPC and DECnet.

Support for SHARP APL and SequeLink Data Direct is available from Soliton Associates 24 hours per day, 365 days per year.

APL2000 Inc (Bloomsbury Software Ltd)

APL2000 Inc is a newly formed entity that has acquired certain APL assets from Manugistics. Among the assets acquired were the PC and UNIX APL products.

This was a strategic acquisition on two levels:

- APL2000's core product — LEX2000 Financial Reporting Software — is built in APL. That gives them a powerful vested interest in ensuring that APL's own development keeps pace with other technologies and with market demands.
- Second they believe there is a need to accelerate APL's evolution to ensure that its capabilities keep pace with other languages. They intend to fill that need.

In addition to that announcement there is good news and bad news.

First the bad news — prices are going up (see *Product Guide* on page 34).

And the good news: they will use the additional revenues to fund an aggressive development effort for APL. APL2000 take the position that because the worldwide universe of APL developers is relatively small, current revenues are not sufficient to support the kind of development required to ensure APL's continued viability in the marketplace. APL products will be priced competitively, but it is vital that they generate enough revenue to support on-going development and evolution.

APL2000 hope users will view the additional cost as an investment in the future of the product. They do, and are committed to providing users a return on that investment. They are assembling the most experienced and qualified team of APL developers in the business.

APL+Link provides the connection between APL+Win and data stored in non-APL databases. It uses ODBC connectivity to access a wide variety of databases on different hardware platforms.

For further information contact Bloomsbury Software on 0171 436 9481.

Insight Systems / Adaytum Software

There is little to report in terms of new products from Insight Systems during the last quarter of 1995. We have ported our products in order to support Dyalog APL Version 8 under Windows 95 and NT, but most of our energies have been spent on version 2.0 of KPS, and growing the company to be prepared to support both KPS and the continued development of our APL Client/Server product line.

1995 has seen Insight Systems more than triple in size from 4.5 persons to 15. The group of APL developers grew from 2 to 6; C developers from 1 to 2. The bulk of this growth happened during the last quarter of the year, coinciding with the rush to meet KPS version 2.0, so this quarter has taught us a new meaning of the word "busy".

Our first major product release in 1996 will be version 2.0 of KPS, Adaytum's flagship product for multi-dimensional business planning. Version 2.0 is the first full Windows version of KPS; a Windows 'Multiple Document Interface' product which will be based on Dyalog APL version 7.2 under Windows 3.1 and OS/2, and version 8.0 under Windows 95 and NT. This and future versions of KPS will rely heavily upon the Client/Server product line, which will also be enhanced significantly during 1996 as a result of our own requirements in addition to those of our customers:

KPS Version 2.0 will make use of the Insight Systems' SQAPL Client product in order to provide read/write access to ODBC data sources. Product releases scheduled for the 2nd and 3rd quarters of 1996 will use the SQAPL Server and SequeLink Client/Server Middleware to provide access to the KPS engine as an ODBC data source.

KPS will use APL Pipes for internal messaging in multi-user environments to provide "groupware" functionality. Finally, the upcoming extension of APL Pipes to support Visual Basic and C DLL Client modules will be used to provide access to the KPS Macro Language from development tools such as Visual Basic, Borland Delphi, or C.

The sales and marketing side of the company has seen the same rate of growth as has development. We have built new sales teams to market KPS and SequeLink in Scandinavia, and continue to market SQAPL in conjunction with SequeLink worldwide. If you are having problems keeping track of this growing portfolio of products and services, remember to set a week aside at the end of July to visit us at APL96 in Lancaster, where we will provide you with the opportunity to see presentations of, or take part in, hands-on workshops using our products.

Dyadic Systems Limited

Dyadic has received an excellent response to its Version 8 Preview Program for Windows 95 which is now closed to new subscribers.

Although Version 8 for Windows 95 and NT is Dyadic's flagship product, the company recognises the need to provide ongoing support and further enhancements for Windows 3.1 and OS/2 users. It has therefore announced Dyalog APL/W Version 7.2 to meet this requirement. Version 7.2 will include all of the enhancements provided in Version 8 with the exclusion of those that rely on features found only in Windows 95 and NT 3.51. Version 7.2 is intended for Windows 3.1 and OS/2 but will also run under Windows 95.

The following enhancements, already finished or under development in Version 8, are just some of the new features that will also be provided in Version 7.2:

- New Native File System (⎕*NCREATE*, ⎕*NTIE*, ⎕*DR* and so forth).
- Control Structures.
- A search path mechanism coupled with a facility to control the export of objects from namespaces. Among other benefits, this avoids the need to specify full pathnames for utilities stored in namespaces.
- Various enhancements to the Grid object including multi-level titles, extended formatting and support for graphics.

- A facility to attach and detach a GUI structure to and from a namespace without destroying any functions and variables it contains. This enhancement simplifies the maintenance of GUI objects as namespaces.
- The capability to draw graphics inside any GUI object that has its own window. This allows you to embellish standard controls such as Buttons and Combo boxes with your own graphics.

Version 7.2 and Version 8 will share a common set of manuals which will be available during the first quarter of 1996. A limited number of copies of version 7.2 will be available for testing during January but upgrade pricing has yet to be finalised.

Causeway Graphical Systems Ltd

Since November, we have been fully occupied in turning Causeway for Dyalog/W into a fully-supportable, professional package suitable for major APL sites to include alongside other development and object-management tools. We have been greatly helped by the use of *namespaces*, as the Causeway class table now becomes a much simpler and more comprehensible structure:

```
#.Gui ---+         ⍝ All the Gui functions are now here
         |
     #.Gui.classes ---+        ⍝ All classes are here
                      |
                  #.Gui.classes.ac     ⍝ Action Button
                      |
                      | functions (such as Disable)
                      | variables (property defaults)
                      | documentation (e.g. design)
```

This makes it possible to view and modify class code with the standard Dyalog editor, which obviously makes classes very much more accessible to APL application developers. It also becomes very easy to integrate this structure with your in-house utility management, as a namespace can be written as a file component, and also retrieved with minimal delay into the workspace.

The `design` variable is another important step towards openness:

```
[General]
name=Action Button
type=Buttons

[Properties]
Appearance=*
Behaviour=*
Accel=txt
```

```
Default=bool
Bitmap=txt
Hint=txt
Tip=txt

[CueCard]
... descriptive text goes here
```

The new Causeway designer groups objects by type (rather than just accumulating new ones under 'Special') and it offers all the allowed properties of an object in a grid structure, similar to Borland's 'Object Inspector' as found in Delphi. You can either give a simple type declaration such as `bool` or `txt` for your property, or you must provide the name of a suitable function. In this case the grid will just have a button to call your own code, which can do whatever is necessary to maintain the property.

This effectively removes the need for object-specific styles in `∆styletab`, leaving only global settings such as font and colour. It also allows all objects to have a local event table, which reduces `∆events` to a much shorter (and unchanging) table of truly generic events such as MouseDown. The effect will be to decouple the Causeway architecture completely from class and application code, allowing us to maintain and improve the `Gui.xxx` functions (including the designer) while users can enhance and modify classes quite independently.

In order to allow for future change, we have restructured the dialogue box definition (hopefully for the last time) as a five-column array. Columns 1-4 remain as they were (Class ID, Caption, Position, Size) and column 5 now contains a set of Property-Value pairs, one pair for each non-null property listed in the class's design documentation.

Clearly, this is much easier to extend, and will allow Causeway to accommodate new concepts such as Hint and Tip cleanly. We will also take the opportunity to fix all the little anomalies such as the ordering of the columns in the event table (see Jan Karman's review) which have accumulated over the years.

Naturally, Causeway II will be shipped with a full set of utilities to find and migrate existing dialogue boxes, and to assist in moving class code from `∆classtab` into the new namespace structure.

We have already shipped a namespace version of the *Rain* graphics software to selected users for testing, and will have this ready to go out with Dyalog 7.2 this month. We are also nearly ready to ship a major new set of printing utilities (working title *NewLeaf*) designed to bring the same quality and ease of use to report-preparation from APL.

THE EDUCATION VECTOR

January 1996

Editor Ian Clark

This Education Vector has been reprinted from VECTOR Vol.12 No.3. VECTOR is the Quarterly Journal of the British APL Association. For more information about the British APL Association, please contact: Sylvia Camacho, 11 Auburn Road, Redland, BRISTOL, BS6 6LS. Tel: 0117-973 0036.

Contents

Ian Clark
IAC/Human Interfaces,
9, Hill End, Frosterley,
Bishop Auckland,
Co. Durham DL13 2SX.

Tel: 01388-527190
Email: 100021.3073@compuserve.com

Editorial

My last editorial ("Raising Standards — or Flying Kites") prompted some Internet correspondence. Since it was much more interesting than anything I could write in this issue, I reprint the most thought-provoking letter, together with my reply (both edited).

(Nick Cox, of Durham University, writes:)

Your opinions do not tally with the evidence known to me or with my experience in what happens to be the university down the road from you. I am taking you seriously and will therefore have misunderstood you if what you intended was merely a provocative musing aloud, somewhat tongue in cheek.

The (quote:) "more sought-after universities" include some of the older traditional universities such as Durham. The evidence coming out from Teaching Quality Assessments is that on the whole these have better teaching than elsewhere, particularly better than the newest universities. Numbers and quality of staff are typically better and resources much better, and at least some staff retain very high dedication to teaching (often at some considerable personal cost to their career advancement). Your innuendo is entirely inaccurate about many people of this kind. It also misses the point that we have a contractual obligation to both teaching and research, not just teaching.

Low correlation between achievement at around 18 and that at around 21 is an important puzzle and I do not have a glib explanation for it. But in effect we are analyzing relatively narrow bands, which raises the possibility of a statistical artefact. If you take height and weight data only for (say) the very tallest, the correlation will be much lower than for the population as a whole. In addition, many students experience very different styles of teaching at university compared with school, and it is hardly surprising that they make different progress and often leap-frog each other. The absorbers at school may be very poor arguers and analysers, to mention just one example. I would be interested in an analysis of performance at 18 compared with 15.

Far from being ruthless to losers, this university devotes considerable time to the idle and those incapable of minimal self-organisation, as well to casualties who deserve the sympathy and consideration they get. Its drop-out rate is accordingly very low. I am not clear whether you would regard that as a good thing.

I find it difficult to reconcile your cynicism about this region only needing one university with your cynicism about excluding a majority of the population from tertiary education. Even if the North-East should turn out to be a net exporter of graduates, that still amounts to a major contribution to the outside world and a major source of employment in the region (not to mention scientific and cultural contributions and so forth). Why sneer at that?

(Ian Clark replies:)

Although you've obviously taken hurt at the things I wrote, I don't essentially disagree with a word you say. Yes, I was being provocative and "tongue in cheek", but I do feel deeply about the things I was writing. So — yes, I want to be taken seriously, but maybe you've got me wrong.

I stand by my central argument that the traditional elitism of Gillian Shephard (the UK Minister for Education) braying about "standards" is an inappropriate answer to the questions posed by poor uptake of university places in science, technology and mathematics. But to answer them needs a book, not an editorial.

Yes, I know all too well how zeal for the quality of one's own teaching, or even worse, trying to improve your school's teaching methods, is to risk your own advancement inside your establishment. My one-time boss, an applied mathematician of the old brigade, once told me that I should be busy enhancing my personal research status and that of the school, not bothering with all that stuff. He was referring to new methods I was promoting on the old HND Computing-A option (which thanks to my human factors research experience I could actually make work). I suppose I upset him by referring to myself as an ex-mathematician — implying that its didactic traditions contributed nothing to the effective uptake of the principles it pretended to deliver. Gauss was called "the old fox who covered his tracks" (meaning that the was careful to expunge from his proofs all evidence of how he arrived at them). His tradition is still upheld. It's not exactly student-friendly, but it certainly cuts a dash.

I can imagine that you, like me, have spent much time and agony over your students. Perhaps more, in hindsight, than some of them deserved. But I resent it only for a very small proportion (my suggested <5%, for which sensible selection methods might possibly be developed to detect). High drop-out rates are often attributed to poor selection procedures. Illogically the converse is often inferred. But I know, and you know, that low drop-out rates have more to do with caring for one's students, than being in the happy position to skim the "cream" of candidates (whatever that is, beyond the flotsam of culture and privilege).

The conscientious teacher of his or her students was the last person I wanted to attack. At Durham, in particular, I have had close contact with staff in many different departments, in a variety of capacities as it happens. Please treat my remarks about what is open the the "more sought-after universities" as describing a standing temptation rather than a pervasive vice, as far as Durham is concerned. If Sodom and Gomorrah had boasted half as many just men the cities wouldn't have got burnt.

Does the North-East (of England) really need all its universities? You're right to perceive a threat here, one which has been uttered in other quarters. But it's really a different question to "does the UK — or the world even — need the N-E universities?" and different again to "do they enhance the region — or the world?"

I've seen the N-E described as a "branch-factory economy". People don't "get on" by moving there and it's not where the big decisions are made. Nationwide organisations treat it as a penal colony for their managers. So how many graduates do the five N-E universities need to produce to fill vacant responsible posts in the N-E? Enough to keep just one of them open? I doubt even that.

But that's not the point. Perhaps I should have italicised "purely to serve the region's needs", but I had hoped that spouting free-market economics ("massive overcapacity...!") in the context of centres of learning and excellence would be instantly recognised as heavy irony. But not as sneering at the region's achievements. I suppose I reckoned without the fact that there are people in power who really do think like Mr Muckybrass. So many that I can virtually lift Rattigan's manifesto out of "Basil the Great Mouse Detective" and people will think I'm not joking.

Some Articles in the Current Vector

Turtle Graphics in J	David Benn
3-Dimensional Noughts and Crosses	Duncan Pearson
Multiprecision Arithmetic (Part 3)	John Sullivan
A Note on Programming Style	Roger Hui and K.E. Iverson

Counting Problem

by Tony Goddard (a.goddard@queens-belfast.ac.uk)

Count different possible colouring of 2 x 2 tiles with n colours, given that rotated or flipped tiles are equivalent. The counts can be computed using two APL functions.

Firstly, define a function *D8* to generate all the rotations and flips of a tile, then define another function *C4T* to look at all of the n*n*n*n tiles, and count only those which are distinct after all rotations and flips are taken into account.

```
   Z ← D8 N;P;Q
 ⍝ Eight rotations
  N ←(N,N)⍴⍳N × N
  Q ← ⌽P ← ⍉N ⋄ Q ←,Q ⋄ P ←,P
  Z ← N ← ⍳⍴P
  Z ← Z,N ← Q[N]
  Z ← Z,N ← Q[N]
  Z ← Z,N ← Q[N]
  Z ← Z,P
  Z ← Z,P ← Q[P]
  Z ← Z,P ← Q[P]
  Z ← Z,P ← Q[P]
  Z ←(8,⍴P)⍴Z

  Z ← C4T N;C;D;E;I4;J;K;N4;U
 ⍝ Colour 2x2 tile with N colours
  I4 ← ⍳N4 ← N × N × N × N
  D ← D8 2
  U ← N4⍴0
  C ← 0
 AA:J ← U⍳0
  →(J ≥ N4)/XX
  C ← C+1
  E ←(4⍴N) ⊤ J
  K ← (4⍴N) ⊥ ⍉E[D]
  U[K]← C                      ⍝ all colourings equivalent to J
  → AA
 XX:Z ← C
```

```
  n  c(n)
  1    1        c(n) = t(t(n)) where t(n) is n th triangular number.
  2    6
  3   21        This looks plausible, but is it true ?
  4   55
  5  120        t(n)= n(n+1)/2
  6  231        t(t(n)) = (1/8) (nn+n)(nn+n+2)
  7  406                 = (1/8) (nn+n)(nn+n)+2nn+2n
  8  666                 = (1/8)  nnnn+2nnn+3nn+2n
  9 1035
 10 1540          & this is a polynomial in n, writing nn for n*n etc.
```

This problem was investigated by Polya. He found a formula, involving the cycle index of a group of permutations. Here the group is known as the dihedral group. It is the group of transformations of the square. The Cycle index is a sum over the elements of the group G:

```
S = (P[1] + P[2] ... + P[m]) / m
```

where m = |G| is the number of elements in G and each P[i] is a monomial in symbols X[1] X[2] ... X[n]. If an element g in G splits into disjoint orbits O1, O2, .. Ok, then it contributes a monomial which is a product from the set X[1], X[2], .. X[n] where an orbit of length k contributes a term X[k]. For example, the identity permutation contributes a term X[1] * X[1] ... * X[1] taken m times.

```
  D8 2     cycles
 0 1 2 3   (0)(1)(2)(3)    X1*X1*X1*X1
 2 0 3 1   (0231)          X4
 3 2 1 0   (03)(12)        X2*X2
 1 3 0 2   (0132)          X4
 0 2 1 3   (0)(12)(3)      X1*X1*X2
 2 3 0 1   (02)(13)        X2*X2
 3 1 2 0   (03)(1)(2)      X1*X1*X2
 1 0 3 2   (01)(23)        X2*X2
```

the cycle index sum is

```
P[G](X1,X2,X3,X4) = (1/8) X1^4 + 2 * X1^2 *X2 + 3*X2^2 + 2 * X4
```

A preliminary result of Polya's theorem is that the number of distinct colourings of the configuration, given n colours is given by

```
|C| = P[G](n, n, ... n).
```

In this particular case, the count is:

```
|C| = (1/8) nnnn + 2nnn + 3nn + 2n
```

References

[1] Fred S. Roberts. *Applied Combinatorics*. Prentice Hall, 1984.

[2] G.Polya. *Kombinatorische Anzahlbestimmungen für Gruppen, Graphen und Chemische Verbindungen*. Acta Math. 68 (1937) 145-254

J-ottings 8

by Norman Thomson

First some statistics. The end of a year is a good time to assess the impact of J in the world, and by a happy coincidence Jan 92 was the first number of "Vector" which carried technical material on J. To assess the strength of J relative to APL, I counted the proportion of technical pages in "Vector" which were devoted to the two languages. I have used the term "technical page" to exclude, for example, advertisements, editorials, product guides, and the social kinds of conference report. While there is inevitably a degree of vagueness in what constitutes a technical page, this does not obscure the broad trends, which the figures below show clearly.

	%age of tech. pages about J				overall %age of J	Total no of technical pages	%age of APL9x conference papers on J
year	Jan	May	Jul	Oct			
1992	42	35	21	24	31	365	3
1993	23	27	0	21	24	296	16
1994	10	35	26	27	23	438	10
1995	21	79	17	42	41	434	28

These figures would suggest that in terms of articulation, J is now about half as strong as APL. A calculation in terms of revenue would probably tell a different story. Some of the early J material was at a rather advanced level, which may help to account for the 0 in Jul 93. It was this which in turn provoked the first of these columns in Oct 93, these being in some sense a plea that "Vector" should be a vehicle for more graduated introductory material on J. Articles such as that by Chris Burke in Vector 12.2 are a sign that "Vector" is indeed becoming a such a medium.

The starting point for the following thoughts is Walter Spunde's article in Vector 12.1 entitled *"J – Where Have All the Variables Gone?"*. As always my goal is to examine J by the morsel rather than the mouthful!

Even the most starry-eyed J enthusiast will probably admit, if pressed, that it is easier to realise a programming idea quickly in APL than in J, and J learners must certainly find themselves in the situation of "I know how to do that in APL, but how do I convert it to J?". Walter in his article effectively does a transcription of the APL expression

```
(↑R) + ((¯1↑R) - ↑R) ÷ L) × ⍳L+1
```

into J, and the solution he arrives at would be essentially the same as that arrived at by any other literal transcriber, namely

```
  {.@] + (({: @] - {.@] )%[)*(i.@([+1:))
```

I will return to this example later with a view to examining alternatives, but first I want to pursue an analogy between transcribing APL to J and making transcriptions in music. The latter occurs when the same music is moved to a different medium, e.g. from symphony orchestra to piano duet. In music, transcriptions are improved by recognition of forms and structures which are inherently characteristic of the medium being transcribed into. In the case of J these are hooks, forks, and sequences as rendered by atop. The fork is probably the single most important form. I find it best to think of it as a central operation applied following two outer transformations, both of which are performed on the combination of left and right arguments — Walter's "invisible" data of which the J programmer must always be keenly aware. The first part of the previous sentence is an expression in words of the J definition of the fork `f g h`, namely:

```
      (x. f y.)  g  (x. h y.)
```

I want to base a second example for transcription on Alan Sykes' article *"Calculating Probabilities for Elementary Distributions"*, which also appeared in Vol. 12 no. 1. In Volume 12 No.2 I proposed the following alternative algorithm for calculating the logarithm of `r!n` :

```
  [0]   Z ← n lnncr r
  [1]   Z ← +/(⍟1+n - ⍳r) - ⍟⍳r←r⌊n-r
```

Examine first what this function does in terms of a simple example, say `6 lnncr 4`. The first step is to transform r into (n-r) if the latter quantity is smaller, so in this case 4 is transformed into 2. The next step is to construct a numerator of two descending integers 6 5, and a denominator of two ascending integers 1 2.

The third stage is divide these by taking logs of each and subtracting, and the final stage is to sum these logs.

Looking at the first step, the data is `r!n` and this has to be transformed to `(n-r)!r`, so that the function min(r,n-r) can be applied. A fork structure is clearly recognisable here with min as the central operation. Of the two preliminary transformations, the rightmost is `-`, the leftmost is "take the right argument", and so the required fork is:

```
  k=.]<.-
```

The pattern "numerator divide-by-subtracting-logs denominator" is another clear fork, of which the central operation is well-known as a J adverbial form namely `-&^.` (see J-ottings 3 for details). We have hereby dealt with both occurrences of log in the APL line, as well as the subtraction. The next step is to seek the two transformations which give the numerator and denominator, bearing in mind that the J function is written in origin 1, whereas J respects only origin 0. Returning to the numerical example, 6 5 is 6 (right argument) - 0 1. `-i.` is a hook which forms the central operation of a fork, of which the left transformation is "right argument", and the right is the transformation already identified as k. Hence define

```
g=.[(-i.)k
```

The transformation defining the denominator is even simpler. In the numerical example, the value required is simply 1 2, that is "increment by 1" (`>:`), following (that is, atop) i., following the transformation k. Hence define

```
h=.>:@i.@k
```

The central fork is now defined, and it remains only to add the `+/` in sequence:

```
lnncr=.+/@(g(-&^.)h)
```

A possible objection to this transcription is that both `g` and `h` contain `k` as their rightmost verb which is both inefficient and inelegant. Whether there is a purely formal way of factoring the k's is an interesting question beyond my powers of analysis. (Perhaps the generalisation of this problem could be suitable material for a Ph.D thesis!). Pragmatically, what has to be observed is that the core fork `g(-%^.)h` is itself the central operation in a broader fork whose right transformation is `k`, and whose left is "leave the left argument as it is". Thus the revised definition, following the factoring of `k`, is:

```
  g1=.-i.
  h1=.>:@i.@]
  lnncr1=.+/@( [ ( g1(-&^.)h1 ) k )
```

Of course it is possible to replace the definitions of `g1`, `h1` and `k` directly to give the sort of one-liner effect which gave APL such a bad name in past times:

```
  lnncr2=.+/@([ ( (-i.) (-&^.) (>:@i.@]) ) k )
```

This is the sort of line which I imagine caused Dick Bowman in his review of J for Windows in the current "Quote Quad" to say that J looks ugly. I find it surprising that punctuation symbols which are totally inoffensive when sprinkled through text should be so unappealing when they appear in a concentrated stream. The moral is simple, namely to break down J verb definitions into bites of about eight characters or so — about the length of an English word which, even if previously unknown to the reader, can be readily assimilated in a single glance. I imagine that few would argue against the superiority of a suitably commented sequence of verbs which should make the algorithm completely clear on subsequent revisiting:

```
k=.]<.-                  NB. transform r of nCr to n-r if smaller
g1=.-i.                  NB. numerator is   n.(n-1). ...
h1=.>:@i.@]                 NB. denominator is 1. 2.  ...
lnncr1=.+/@([(g1(-&^.)h1)k)  NB. sum logs following subtraction
```

As a final comment cap (`[:`) can be used instead of atop. The effect is to make the outer transformations on forks apply monadically. For example `h1` above could be written equivalently as

```
h1=.[:>: [:i.]
```

Whether atop or cap is preferable in such situations is a matter of personal taste. I am happy to think in terms of sequences passing results leftwards which produces the monadic effect in another way.

Returning to Walter's example, there is a clear primary fork which adds the start value to all items in an arithmetic sequence of length n+1. What is added to a is a division of the range (b-a) by n, multiplied by 0,1, .. n+1, so choose the second major fork to have `%` as its central operation. The rest of the analysis (p and q below) is what is obtained as answers to the question: "What are the transformations required to produce the two parts of this quotient?". Here is my solution:

```
p=.i.@>:@[               NB. 0,1, .. n
q=.%-/@-                 NB. reciprocal of range
r=.{.@]+p%q              NB. a + (0,1/n, .. 1)*range
```

Comparison with Walter's version requires substituting for `p` and `q` in `r`, which gives:

```
r1=.{.@]+(i.@>:@[%(%-/@-))
```

This has 22 characters as opposed to Walter's 32, and there are only two references to `[` and `]`, as opposed to five in Walter's. The overall moral is: be fearless in transcribing from APL to J, but be sensitive also to the receiving medium. The question of formal analysis and manipulation of J expressions is intriguing, but hard!

Towers of Hanoi, Simplified in J

by Howard A. Peelle
Email: hapeelle@educ.umass.edu

Introduction

Towers of Hanoi is a classic puzzle comprised of three pegs and a stack of different-sized discs, as depicted below in its starting state with 4 discs:

```
           ||                 ||                ||
          ====                ||                ||
         ======               ||                ||
        ========              ||                ||
       ==========             ||                ||
 ---------------------------------------------------------
          START              PARK              GOAL
```

The objective is to get all discs onto the GOAL peg by moving one disc at a time, using the PARK peg temporarily, without ever placing a larger disc on top of a smaller disc.

The general solution (for N discs) is well-known:

Move N-1 discs to the PARK peg
Move the Nth disc to the GOAL peg
Move N-1 discs to the GOAL peg

This recursive algorithm specifies how to solve the N-disc puzzle in terms of the (N-1)-disc puzzle (twice), which uses the same algorithm for one less disc with PARK as GOAL (in the first step) and with START as PARK (in the last step), both of which in turn use the same algorithm for one less disc, etc. Ultimately, the 1-disc puzzle is easily solved by just moving one disc (and no move is needed for 0-discs).

Accordingly, the Towers of Hanoi puzzle is often used to illustrate recursion, e.g. [1] in APL, [2] in APL2, and [3] in J. It is typically programmed either by representing *pegs* or by representing *discs* or both. (See Appendix.) Here, however, we will represent *moves* for the Towers of Hanoi, which enables a simplified program definition, as well as a mnemonic solution.

Moves for Towers of Hanoi

The three basic ways to move a disc between pegs are named (arbitrarily) A, B, and C, as shown below:

```
                         B
      <----------------------------->
  ||                    ||                    ||
  ||         A          ||         C          ||
  ||   <----------->    ||   <----------->    ||
  ||                    ||                    ||
  ||                    ||                    ||
------------------------------------------------
 START                 PARK                  GOAL
```

Note that although a move may occur in either direction, only one will be legal since the other direction would place a larger disc on top of a smaller one. So there is no ambiguity in this representation.

Program to Solve Towers of Hanoi

Here is a generalized J program to generate a sequence of moves to solve the Towers of Hanoi puzzle for any number of discs.

It uses the following utility conjunctions for ease in reading:

```
  Else =. `   NB. Else is Tie, which ties two verbs together
  If =. @.    NB. If is Agenda, which selects first verb
              NB. on left if verb on right returns 0 (false)
              NB. or second verb if 1 (true)
```

The main program requires two inputs: the names of the moves (corresponding to the diagram above) and the number of discs (a positive integer). Its definition is:

```
Hanoi =. ((Park Hanoi ]-1:),Move,(Goal Hanoi ]-1:)) Else Move If (]=1:)
  NB. Hanoi is a recursive program which will call itself
  NB. (twice) or else just do subprogram Move
  NB. if the right input (]) equals 1.
  NB. Each recursive part executes Hanoi with names of
  NB. moves in a new order (from subprogram on its left)
  NB. and one less number of discs (on its right),
  NB. then the results are joined with Move.
```

As specified in the first step of the algorithm (above), subprogram `Park` exchanges the first two moves (e.g. A and B) in order to move N-1 discs from the START peg to the PARK peg (instead of the GOAL peg):

```
Park =. (1:,0:,2:) { [      NB. Second, first, third items
                            NB. From Left input
```

As specified in the second step of the algorithm, subprogram `Move` returns the second move (e.g. B) which moves the largest disc to the GOAL:

```
Move =. 1: { [              NB. Second item From Left input
```

And, as specified in the last step of the algorithm, subprogram `Goal` exchanges the last two moves (e.g., B and C) in order to move N-1 discs from the PARK peg (instead of the START peg) to the GOAL peg:

```
Goal =. (0:,2:,1:) { [      NB. First, third, second items
                            NB. From Left input
```

Examples

```
   'ABC' Hanoi 1        NB. Solve Hanoi with moves 'ABC'
B                       NB. for 1 disc
   'ABC' Hanoi 2        NB. Solve for 2 discs
ABC

   'ABC' Hanoi 3        NB. 3 discs
BACBACB

   'ABC' Hanoi 4        NB. 4 discs
ABCABCABCABCABC

   'ABC' Hanoi 5        NB. 5 discs
BACBACBACBACBACBACBACBACBACBACB
```

Mnemonic Solution

Notice the patterns in the move sequences. For an odd number of discs, the solution is a sequence of repeated BAC moves with a final B. For an even number of discs, the solution is a sequence of repeated ABC moves. Since the total number of moves for N discs is `<:2^N` (one less than 2 to the Nth), the solutions are easy to remember. Or, in practice, just repeat the move sequence ABC (for even N) or BAC (for odd N) until the puzzle is solved.

Note that any names may be used for the moves. For instance:

```
   ('hop';'skip';'jump') Hanoi 3        NB. Words as moves
                                        NB. for 3 discs
┌────┬───┬────┬────┬───┬────┬────┐
│skip│hop│jump│skip│hop│jump│skip│
└────┴───┴────┴────┴───┴────┴────┘

   1 2 3 Hanoi 4                        NB. Numbers as moves
1 2 3 1 2 3 1 2 3 1 2 3 1 2 3           NB. for 4 discs
```

Further, notice the symmetry in the solutions. The second half of the move sequence is identical to the complement of the first half reversed. In other words, the sum of the entire sequence and its reverse is all constants:

```
   (+ |.) 1 2 3 Hanoi 4             NB. Result Plus Reverse result
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
```

Efficient Program for Towers of Hanoi

The above pattern suggests a more efficient program:

```
  Hanoi =. {~ I             NB. Hanoi is (I right input)
                            NB. From (left input)
  I =. M $ P A. i.@3:       NB. I is M Reshape Pth Atomic
                            NB. Permuted Integers 0 1 2
                            NB. 0 A. 0 1 2 is 0 1 2 and
                            NB. 2 A. 0 1 2 is 1 0 2
  P =. +: @ (2&|)           NB. 0 if discs are even or
                            NB. 2 if discs are odd
  M =. <: @ (2&^)           NB. Number of moves in solution
```

Results are the same as for all examples above.

Howard A. Peelle
University of Massachusetts
Amherst, MA 01003 USA

References

[1] Peelle, H.A. *APL: An Introduction*, Holt, Rinehart & Winston, New York, NY 1986 pp. 308–309

[2] Brown, Pakin, & Polivka *APL2 At a Glance*, Prentice-Hall, Englewood Cliffs, NJ 1988 pp. 321–323

[3] Iverson, K. *J Introduction and Dictionary*, ISI, Toronto, CA 1994 p. 23

Appendix: Other Programs for Solving Towers of Hanoi

Peg Representation in APL (from [1]):

```
      ∇ PEGS HANOI N
[1]    →0 IF N=0
[2]    PEGS[1 3 2] HANOI N-1
[3]    'MOVE FROM ',PEGS[1],' TO ',PEGS[3]
[4]    PEGS[2 1 3] HANOI N-1
      ∇

      ∇ B ← L IF C
[1]    B ← C/L
      ∇

    'ABC' HANOI 3    ⍝ Pegs named 'ABC' for START,PARK,GOAL
MOVE FROM A TO C
MOVE FROM A TO B
MOVE FROM C TO B
MOVE FROM A TO C
MOVE FROM B TO A
MOVE FROM B TO C
MOVE FROM A TO C
```

Peg and Disc Representation in APL2 (from [2]):

```
[0]   N HANOI NEEDLE
[1]   →(N=0)/0
[2]   (N-1) HANOI NEEDLE[1 3 2]
[3]   'MOVE DISK' N 'FROM' NEEDLE[1] 'TO' NEEDLE[2]
[4]   (N-1) HANOI NEEDLE[3 2 1]

    3 HANOI 1 2 3      ⍝ Pegs named 1 2 3 for START,GOAL,PARK
MOVE DISK 1 FROM 1 TO 2    ⍝ Disks names 1 2 3 ...
MOVE DISK 2 FROM 1 TO 3
MOVE DISK 1 FROM 2 TO 3
MOVE DISK 3 FROM 1 TO 2
MOVE DISK 1 FROM 3 TO 1
MOVE DISK 2 FROM 3 TO 2
MOVE DISK 1 FROM 1 TO 2
```

Peg Representation in J (from [3]):

```
h=. b`(p,.q,.r)@.c
c=. 1: < [
b=. 2&,@[ $ ]
p=. <:@[ h 1: A. ]
q=. 1: h ]
r=. <:@[ h 5: A. ]
```

```
3 h 'ABC'         NB. Pegs named 'ABC' for START,GOAL,PARK
AABACCA
BCCBABB
```

Alternative: Report only pegs to move to and always move (legally) the smaller disc not just moved

```
h =. # ` (p , m , r) @. c
c =. [ > 0:
m =. 1: { ]       NB. p and r same as above

3 h 'ABC'         NB. Pegs named 'ABC' for START,GOAL,PARK
BCCBABB           NB. Pegs to move to
```

Disc Representation in J:

```
H =. (H , >: , H)@<: ` i. @. (< 1:)
H 4                 NB. Discs named 1 2 3 4 1 2 1 3 1 2 1 4 1 2
1 3 1 2 1           NB. Incomplete solution
                    NB. (where to move to?)
```

Alternative using Power conjunction:

```
H =. ] , (1: + >./) , ]
H^:3 (1)
1 2 1 3 1 2 1 4 1 2 1 3 1 2 1
```

Alternative using binary array:

```
binary =. #: @ i. @ (2&^)
change =. }. ~: }:
H =. +/"1 @ change @ binary
H 4
1 2 1 3 1 2 1 4 1 2 1 3 1 2 1
```

Complete solution:

```
H =. onto @ move"1 @ }. @ binary
binary =. #: @ i. @ (2&^)
move =. +/\@|. >:@i. 1:,2:
onto =. {:@sign 1} ]
sign =. -`+ @. odd =. 2&| @ diff =. -/

   |: H 4
 1 2 1  3  1 2 1 4 1  2 1 3  1 2 1    NB. Disc to move onto
_5 5 2 _5 _3 3 2 5 4 _4 2 4 _3 3 2    NB. or not (_) onto
                                      NB. another disc
                                      NB. (n+1 is empty peg)
```

APL Product Guide

compiled by Gill Smith

VECTOR's exclusive APL Product Guide aims to provide readers with useful information about sources of APL hardware, software and services. We welcome any comments readers may have on its usefulness and any suggestions for improvements.

Pressure on space occasionally prevents us from printing the complete guide, however updates will always be listed. We do depend on the alacrity of vendors to keep us informed about their products. Anyone who is not included in the Guide should contact me to get their free entry — see address below.

We reserve the right to edit material supplied for reasons of space or to ensure a fair market coverage. The listings are not restricted to UK companies and international suppliers are welcome to take advantage of these pages.

For convenience to readers, the product list has been divided into the following groups ('poa' indicates 'price on application'):

- Complete APL Systems (Hardware & Software)
- APL Interpreters
- APL-based Packages
- APL Consultancy
- Other Products
- Overseas Associations
- Vendor Addresses
- World Wide Web and FTP Sites

Every effort has been made to avoid errors in these listings but no responsibility can be taken by the working group for mistakes or omissions.

We also welcome information on APL clubs and groups throughout the world.

All contributions and updates to the APL Product Guide should be sent to Gill Smith, at Brook House, Gilling East, York, YO6 4JJ. Tel: 01439-788385, Email: 100331.644@Compuserve.com

COMPLETE APL SYSTEMS

COMPANY	PRODUCT	PRICES(£)	DETAILS
Dyadic	IBM RS/6000 MD320	11,736	APL POWERstation (Greyscale) 27.5 MIPS, 7.4 Mflops RISC Processor 8Mb RAM, 120Mb Disk 19" 1280x1024 Greyscale Graph Display AIX, OSF Motif, Dyalog APL (1-user)
	IBM RS/6000 MD320	13,817	APL POWERstation (Colour) 27.5 MIPS, 7.4 Mflops RISC Processor 8Mb RAM, 120Mb Disk 16" 1280x1024 Colour Graphics Display AIX, OSF Motif, Dyalog APL (1-user)
	IBM RS/6000 MD320	22,656	Advanced APL POWERstation 27.5 MIPS, 7.4 Mflops RISC Processor 16Mb RAM, 320Mb Disk, 150Mb Tape 16" 1280x1024 Colour Graphics Display AIX, OSF Motif, Dyalog APL (1-user)
	IBM RS/6000 MD520	37,114	APL POWERsystem (8-users) 27.5 MIPS, 7.4 Mflops RISC Processor 16Mb RAM, 320Mb Disk, 150Mb Tape CD-ROM Drive, 16 Ports AIX, Dyalog APL (2-8 user licence)
	IBM RS/6000 MD530	72,054	APL POWERsystem (16-users) 34.5 MIPS, 10.9 Mflops RISC Processor 32Mb RAM, 1.34Gb Disk, 2.3Gb Tape CD-ROM Drive, 16 Ports AIX, Dyalog APL (8+ user licence)
	IBM RS/6000 MD540	122,842	APL POWERsystem (32-users) 41 MIPS, 13 Mflops RISC Processor 64Mb RAM, 1.7Gb Disk, 2.3Gb Tape CD-ROM Drive, 32 Ports AIX, Dyalog APL (8+ user licence)
Interprocess Systems	APL2 Dev't Workstation	poa	Mainframe APL2 supported on a PS/2 via a co-processor card with 16Mb of memory running VM/ESA (370 mode). A complete system includes a PS/2, a P/370 co-processor card, and software licenses for VM/ESA, APL2, GDDM and the full line of Interprocess APL2 enhancements.
Optima	IBM Compatible	poa	Complete PC-based station, APL interpreters & all support eq't

APL INTERPRETERS

COMPANY	PRODUCT	PRICES(£)	DETAILS
APL Software	APL*Plus/PC Release 10	450	STSC's APL for IBM PCs & compatibles.Upgrades from earlier releases also available.
	Run-time	poa	Closed version of APL*Plus/PC which prevents user exposure to APL.
	APL*Plus II	1,395	All the features of mainframe APL*Plus for your 386PC!
	Run-time	poa	
	Dyalog APL	1000-10,000	2nd generation APL for Unix systems
	APL2/PC	poa	IBM's APL 2 for the PC.
Atlantis Software	Analytic Platform (*K*)	poa	*K* is an APL-like language
Beautiful Systems	Dyalog APL/W for Windows	poa	US Distributor of Dyalog APL products from Dyadic.
	Dyalog APL for Unix	poa	See Dyadic listing for product details.
The Bloomsbury Software Company	APL+PC Version 11	260	Upgrade to version 11 gives free runtime (£130 from any vn).
	APL+Win	1400	A 32-bit Windows-hosted interpreter that runs under all Windows platforms including Windows 95.
	Migration to APL+Win	650	from APL*PLUS II versions 4/5
		800	from earlier versions of APL*PLUS II
	APL+DOS	1300	APL*PLUS II DOS is renamed to APL+DOS.
	Migration to APL+DOS	630 / 430	from APL*PLUS/PC or APL*PLUS II

	APL*PLUS II for UNIX	poa	STSC's 2nd generation APL for all major Sparc and Risc Unix workstations.
	APL*PLUS VMS	poa	2nd generation APL for DEC VAX computers under VMS.
	APL*PLUS Mainframe	poa	Enhances VS APL with many high performance, high productivity features. For VM/CMS and MVS/TSO offers simple upgrade from VS APL
Dyadic	Dyalog APL for DOS/386	995	Second generation APL for DOS.Runs in 32-bit mode, supports very large workspaces. Unique "window-based" APL Development Environment and Screen Manager. Requires 386/486 based PC or PS/2, at least 2Mb RAM, EGA or VGA, DOS 3.3 or later.
	Dyalog APL/W for Windows	995	As above, plus object-based GUI development tools. Requires Windows 3.0 or later.
	Dyalog APL for Unix	995-12,000	Second generation APL for Unix systems. Available for Altos, Apollo, Bull, Dec, HP, IBM 6150, IBM RS/6000, Masscomp, Pyramid, NCR, Sun and Unisys machines, and for PCs and PC/2s running Xenix or AIX. Oracle interface available for IBM, Sun and Xenix versions.
IAC/Human Interfaces			
	I-APL/Mac	13	Macintosh version of I-APL
I-APL Ltd	I-APL/PC or clones	8	ISO conforming interpreter. Supplied only with manual (see 'Other Products' for accompanying books).
	I-APL/BBC Master	8	As above
	I-APL/Archimedes	8	As above
	Strand Software Inc		Strand Software Inc has the sole selling rights to Iverson Software Inc products. I-APL stocks a few of these (mainly APLIWIN and the personal J products and books), but is no longer an agent
IBM APL Products	TryAPL2	free	APL2 for educational or demonstration use. Write, fax or Email to APL Products; specify disk size desired.
	APL2 PC (US Version)	$630	Product No. 5799-PGG. PRPQ Number RJ0411. Order from 1-800-IBM-CALL
	APL2 PC (European Version)	£348	Product No. 5604-260. Part number 38F1753. From all IBM dealers, including MicroAPL
	APL2 for OS/2 Entry Edition	$185	Part No 89G1556.
	APL2 for OS/2 Advanced Edition	$650	Part No 89G1697. Contains all facilities of the Entry Edition plus: DB2 interface; co-operative processing TCP/IP interface; tools for writing APs; TIME facility
	APL2 for Sun Solaris	$1500	Product No. 5648-065.
	APL2 for AIX 6000	poa	Product No. 5765-012.
	APL2 Version 2	poa	Product No. 5688-228. Full APL2 system for S/370 and S/390
	APL2 Application Envt Vn2	poa	Product No. 5688-229. Runtime environment for APL2 packages
Insight Systems	APL*PLUS/PC	poa	APL systems marketed and supported ...
	Dyalog APL	poa	from: Dyadic, Manugistics, IBM
	APL2	poa	under: Windows, OS2 and Unix
Iverson Software Inc.	J Professional (inc runtime)	$495	
	J Personal Edition	$100	
	J Personal (disks only)	$40	The text of the manuals is available as Windows Help, so the paper copies are no longer a necessity.
	APLIWIN	$30	For 386/PC under Windows 3.1
	APL Reference Manual	$30	Documentation for all the above.
	J System Kit	$24	J 6.2 diskette with manual "J:Introduction and Dictionary"
	J Source Code	$90	Full C source code plus 100-page book
MasterWork Software	Manugistics Products and ISI	poa	New Zealand distributor

MicroAPL	APL.68000 Level I	2000	First generation APL with numerous enhancements. Multi-user version (Unix, Mirage, MCS).
	APL.68000 Level II	2500	Second generation APL. Nested arrays, user defined operators, selective specification etc. Multi-user version (Unix, Mirage, MCS)
	APL.68000/X	1500-6000	Second-generation APL. Nested arrays, user defined operators, selective specification, etc. Multi-user AIX version with full OSF/Motif support.
	APL.68000 Level I Mac, ST, Amiga	87	First generation APL. Single user, full windowing interface, software floating point support.
	Mac, Amiga	260	First generation APL. Single user, full windowing interface, hardware floating point.
	APL.68000 Level II ST	170	Second generation APL. Full windowing interface, software floating point support.
	Amiga	260	Second generation APL. Full windowing interface.Hardware and software floating point support.
	Mac	520	Second generation APL. Full windowing interface.Hardware and software floating point support.
	APL*PLUS Rel 10 APL*PLUS II V 4.0	450 1395	
Oasis Systems	Dyalog APL	poa	Dyadic Systems
	APL*PLUS	poa	Manugistics
	APL.68000	poa	MicroAPL Ltd
	APL2	poa	IBM
Optima	APL*PLUS/PC	369	
	APL*PLUS II APL*PLUS II PC Developers Kit Dyalog APL	950 poa 999	
RE Time Tracker Oy	APL*PLUS/PC	poa	Complete APL*PLUS and Statgraphics product range and user support for Finland
	APL*PLUS II/DOS		
	APL*PLUS III/WIN		
	APL*PLUS/UNIX		
Soliton Associates	SHARP APL for MVS	poa	for IBM MVS mainframes
	SHARP APL for Unix	poa	for IBM RS/6000 and Sun SPARC
Strand Software	Canada		
	All APL*PLUS Products	poa	All APL*PLUS products including upgrades and educational.
	Dyadic and ISI products	poa	
	USA		
	Dyadic and ISI products	poa	
Uniware	APL*PLUS/PC	495	STSC's full feature APL for IBM PC/XT/AT, Compaq, Olivetti.
	Run-Time	call	Closed version of APL*PLUS/PC which prevents user exposure to APL.
	APL*PLUS/UNX	call	STSC's full feature APL for UNIX based computers
	APL*PLUS II	call	STSC's full feature APL for 386 machines.

APL PACKAGES

COMPANY	PRODUCT	PRICES(£)	DETAILS
Adaptable Systems	FLAIR	poa	Finite loader and interactive rescheduler. Customisable full-function scheduling system. (Available outside Australia by special arrangement only.)
APL-385	APL-385 for APL*PLUS/PC	50	including ...
	FSM-385		Screen development
	DRAW-385		Screen design
	DB-385		Relational W.S.
	GEN-385		Miscellaneous Utilities
The APL Group	Qualedi	$1500-4000	Electronic Data Interchange (EDI) translation software for the PC, with strict compliance checking.
APL Software Ltd (mainframe)	RDS	poa	Relation Data Base System
	IPLS	poa	Project Management System
	REGGPAK	poa	Regression Analysis Package
(microcomputer)	POWERTOOLS	295	Assembler written replacement function for commonly used CPU-consuming APL functions, includes a Forms Processor.
	REGGPAK	poa	Regression Analysis Package
	RDS	990	Relational Database System
Beautiful Systems	ASF_FILE	$399	Dyalog APL/W auxiliary processor for access to APL*PLUS/PC APL component files (*.ASF).
	NAT_FILE	$299	Dyalog APL/W auxiliary processor which emulates the APL*PLUS/PC quad-N native file subsystem for access to the DOS file system.
	DBF_FILE	$299	Dyalog APL/W auxiliary processor for efficient block mode access to dBASE format files. Designed to get large amounts of data in and out of dBASE. Not suited for random access to small amounts of data (it does not handle keys).
The Bloomsbury Software Company (for VSAPL)	Enhancements & Sharefile	poa	Component files, quad-functions & nested arrays for VSAPL under VM/CMS & MVS/TSO
	Compiler	poa	The First APL compiler!
(for APL2)	Sharefile/AP	poa	STSC's shared access component file system for APL2. Comparable to all APL*PLUS file systems: multi-user storage of APL2 arrays with efficient disk usage.
Causeway	Causeway for Dyalog/W	$50	Manuals and Class-management utilities for the Causeway platform supplied free with Dyalog APL/W
	Causeway for APL*PLUS III	$50	Software, class-management utilities and printed documentation for Causeway under APL*PLUS III
	Rain Graphics Workspace	$250	Full on-line documentation (Windows Help) and handy-reference card for the *Rain* business and statistical graphics workspace supplied with Dyalog/W.
Cinerea AB	ORCHART	250	Organization chart package for IBM APL2/PC. Full & heavily commented source code included - free integration into other applications. NB: ASCII output with line-drawing (semi-graphic) characters for boxes.
CODEWORK	HELM	poa	Decision Support system for top management. Developed in Italy over 7 years. Requires APL mainframe or APL*PLUS/II. Optional modules: EIS, Excel interface, DTP output via LATEX, output on map background.
CYBEX AB	APL Graf/PC	290	Presentation graphics for APL*PLUS/PC (CGI)
	APL Graf II/PC	390	Presentation graphics for APL*PLUS II/PC (CGI).
	Utility Functions APL2	1900	For APL mainframe; incl. a very fast search.
	Utility Functions II/PC	130	Same package for APL*PLUS II/PC.

H.M.W.	4XTRA	poa	Front-end Foreign Exchange dealing / pos keeping
	Arbitrage	poa	Arbitrage modelling
	Basket	poa	Basket currency modelling
	Menu-Bar	poa	pull-down menu for APL*PLUS/PC
HRH Systems	APL Utilities	poa	Software to transfer workspaces between APL*PLUS and Sharp, and between APL*PLUS and I-APL. Software to import IBM .ATF files to APL*PLUS.
	APL*PLUS Utilities		Public domain software, unlock locked fns, a user-friendly alternative to locking, fns of mathematical physics, menus, and others.
IAC/Human Interfaces	IAC/Graf	15	Graph plotting for I-APL/Mac
	IAC/Vox	15	Spoken APL characters for I-APL/Mac
I-APL Ltd	Educational workspaces	5	PC format disks with the examples from: Thomson, Espinasse (Kits 1-4), Kromberg, Jizba & FinnAPL. All the examples to save your fingers!
IBM APL Products	A Graphical Statistical System	$250	for DOS, Product Number 5764-009
	(AGSS)	$500	for Workstations (OS/2, Aix, Solaris), Product Number 6764-092
		$2500	for CMS, Product Number 5764-011
Impetus Ltd	*Impetus*	poa	Corporate Modelling and Reporting System.
INFOSTROY	APL*PLUS/Xbase Interface (II/386 Version 2)	$198	Complete package written in C. Comparable with the data, index & memo files of FoxPro, dBASE, & Clipper. Multi-user support. No DBMS license required.
	(PC Version 2)	$98	As above for APL*PLUS/PC.
	(DLL Version 1)	$198	The same in a DLL form! Gives your Windows applications all advantages of DLLs.
Insight Systems	IUTILS/XP	20-95	Cross-platform utility library including simple OS calls (DIR, COPY, DEL, RENAME) and DATE functions. For APL*PLUS II, APL2 and Dyalog APL under Windows, OS/2 and Unix.
	ASI	95	APL Spreadsheet Interface. "Device-independent" spreadsheet driver supporting Excel, 123 and Quattro-Pro for Dyalog APL/W
	WinCom	95	Asynchronous comms package for Dyalog APL/W
	S2D,22D,X2X	poa	Advanced APL syntax analysis and conversion packages from Sharp and APL2 to Dyalog, and between any two APLs
	SQAPL Client	poa	Interface from APL*PLUS II, APL2 and Dyalog (Windows, OS/2 or Unix) to most SQL databases over most networks.
	SQAPL Server	poa	Makes APL*PLUS II, APL2 or Dyalog APL (Unix) available as SequeLink servers. Can be called from SQAPL clients or other applications such as Excel, C++, Smalltalk, Visual Basic.
Interprocess Systems	APL2 Development Workstation	poa	
	IEDIT	$3000-5000	Full screen APL2 editor with immediate APL execution, and full-screen debugger
(mainframe)	AFM	$15300	High performance component and keyed file system (VS APL and APL2)
	Enhanced Format	$2575	A QuadFMT data formatter for VS APL and APL2
	PowerCode	$2000	External functions for APL2
	WSORG	$1500	Full-screen Workspace Organizer for APL2.
JAD Software	JAD SMS	150-500	Software management system for APL*PLUS II based on hierarchical databases; includes full-screen interface and stand-alone functions. Price depends on number of users.
Lingo Allegro	FRESCO Business Graphics	poa	Fast and Easy Business Graphics DLL
	AP126/PC	poa	GDDM Interface for Dyalog APL/W
	AP127/PC	poa	ODBC Interface for Dyalog APL/W

	AP119/PC	poa	TCP/IP interface for Dyalog APL/W
	FACS	poa	EMMA-like interface to DB2 or ODBC databases
	TOPR	poa	APL Code and Application Management for Dyalog APL/W
Mercia	LOGOL 92	poa	Logistics management system for 386/486 & RISC computers. Sales Forecasting, Inventory Management, Master Scheduling, Distribution Requirements Planning, Sales & Operations Planning.
	TWIGS	poa	A modular library of tools to teach and explore state-of-the-art materials management concepts. Developed by R.G. Brown.
RE Time Tracker Oy	UIT/W	poa	TMT-Team Oy's User Interface Toolkit for APL*PLUS II and PLUS III under Windows. Comprehensive spreadsheets, replicated fields, special field types, etc.
	DB+	poa	TMT-Team Oy's database interface for APL*PLUS II & PLUS III under Windows. Interfaces to almost twenty different databases.
Soliton Associates	LOGOS	poa	Application Development Environment
	MAILBOX	poa	Electronic Mail
	VIEWPOINT	poa	Report generator with interfaces to DB2 and MVS data
UNIWARE (for mainframe)	STSC's ENHANCEMENTS	poa	Quad-functions & nested arrays for IBM VSAPL
	STSC's SHAREFILE	poa	component files for IBM VSAPL and for IBM APL2
	TOOLS & UTILITIES	poa	Including FILEPRINT, FILESORT, FILECONVERT FILEMANAGER(EMMA) STSC's database package
	EXECUCALC	poa	Mainframe spreadsheet compatible with VISICALC and part of LOTUS 1-2-3 under VSAPL(VM or TSO)
(for APL*PLUS/PC)	APL Debugger 2.1	FF1950 FF9750	A visual APL debugger to help develop applications (site license)
	Menus 3.0	FF2450 FF12250	Complete set of hierarchical menu utilities (site license)
	ETATGEN 2.0	FF1950 FF9750	Page layout report generator (site license)
	UNITAB 2.0	FF4550 FF22750	An APL*PLUS spreadsheet-like data entry and validation system (site license)
	UNIASM 3.0 (site license)	FF4950	Assembler utilities to speed up APL*PLUS/PC applications
	UNISTAT 5.1	FF2900	Data analysis add-on module for Statgraphics
(for APL*PLUS II)	UNIWARE Toolkit II 4.1	FF39000	(site license only). Relational database system and complete set of utilities for APL*PLUS II development
	APL Debugger II 2.1	FF2950 FF14750	A visual APL debugger to help develop applications (site license)
	Menus II 4.0	FF3950 FF19750	Complete set of hierarchical mouse-driven menu utilities (site license)
	ETATGEN II 2.0	FF2950 FF14750	Page layout report generator (site license)
	UNITAB II 2.0	FF6950 FF34750	An APL*PLUS spreadsheet-like data entry and validation system (site license)
	UNISTAT Plus 5.2	FF4300	Data analysis add-on module for Statgraphics
Warwick University	BATS	250	Menu driven system for time series analysis and forecasting using Bayesian Dynamic modelling. Price is reduced to £35 for academic institutions.
	FAB	free	Training program for the above.
Zark	APL Tutor (PC)	$299	APL computer-based training. Available for APL*PLUS PC & APL*PLUS II. Demo disk $10.
	APL Tutor (MF)	$5000	Mainframe version.
	Zark ACE	$99	APL continuing education. APL tutor news and hotline phone support.

	APL Advanced Techniques....	$59.95	488pp. book, (ISBN 0-9619067-07) including 2-disk set of utility functions (APL*PLUS PC format).
	Communications	$200 pc, $500 mf	Move workspaces or files between APL environments.

APL CONSULTANCY

COMPANY	PRODUCT	PRICES(£)	DETAILS
Active Workspace	APL Programming	poa	Short or long-term consultant available.
Adfee	Consultancy	poa	Development, maintenance, conversion, migration, documentation, of APL products in all APL environments
Andrews	Consultancy	poa	APL programming and analysis, specialises in tree-processing algorithms.
APL People	Consultancy	poa	Consultants available at all levels. Expertise in APL system design, project management, prototyping, financial applications, decision support systems, MIS, links to non-APL systems, documentation, etc.
Bloomsbury Software	Consultancy	300-750+VAT	
Camacho	Consultancy	poa	Manuals; feasibility reports and estimates; analysis and programming; APL and MS Windows applications; Sharp, ISI APL, APL*PLUS, APL2/PC and other APLs spoken. Fixed price systems a speciality
Ray Cannon	Consultancy	poa	APL, C, Assembler, Windows, Graphics: PC and mainframe
Causeway	Consultancy and Training	poa	Management and upkeep of the Causeway development environment for individuals and large APL sites. On-site training for Causeway/Dyalog and Causeway/Plus III
Paul Chapman	Consultancy	poa	24-hr programmer: APL, C, SmallTalk, Graphics; PC, mini, mainframe and network.
David Crossley	Consultancy	poa	Broad experience in many APL environments
Peter Cyriax	Consultancy	100-150 120-200 160-300	Junior Consultant Consultant Senior Consultant
Dogon Research	Consultancy	poa	APL Systems consultancy, design, implementation, support, documentation and maintenance. All dialects with special emphasis on APL2 and Dyalog APL/W.
Dyadic	Consultancy	poa	APL and Unix system design, consultancy, programming and training.
E & S	Consultancy	poa	System prototyping: all types of information system, engineering software, graphics and decision support systems APL*PLUS/PC, APL2, Dyalog APL
Evestic AB	Consultancy	poa	Excellent track record from 10+ years of APL applications in banking, insurance, and education services. All dialects, platforms and project phases. SQL expertise.
General Software	Consultancy	from 120	
Greymantle Assoc	Consulting	poa	Company reporting, business graphics, Windows applications with Dyalog APL/W.
H.M.W.	Consultancy	poa	System design consultancy, programming. HMW specialize in banking and prototyping work.
Michael Hughes	Consultancy	poa	Consultant with 10+ years experience with various APL interpreters and C.
IAC/Human Interfaces	Consultancy	350	APL on Macintosh & PC. HCI design. VDU ergonomics: EC/Health & Safety compliance.
	Documentation	100-200	On-line assistance, product demos & mock-ups, manual writing; foreign language software localization.
	Training	poa	Using I-APL for courseware & distance learning materials; Mac programming in C, APL & HyperCard.

INFOSTROY	Consultancy	poa	APL*PLUS & Windows consultancy. Porting of software written in C into APL*PLUS.
Insight Systems	Consultancy	poa	Experts in APL conversions between any combination of: APL*PLUS, APL2, Dyalog APL and Sharp APL. We are also experienced right-sizers, comfortable with networks and relational databases (that also means when NOT to use SQL) and client/server development in APL, C and Visual Basic.
Intelligent Programs	Consultancy	poa	Systems development, enhancements, support.
	Documentation	poa	Preparation of new manuals, rewriting of existing materials.
	Training	poa	Training for APL experts through to non-technical system users.
JAD Software	Consultancy	poa	Systems design and development, project management, technical manuals, financial and actuarial expertise in APL.
Kestrel	Consultancy	poa	All APLs, all environments. Design, analysis, coding, maintenance, documentation, training, interfacing.
Lingo Allegro USA	Consultancy	poa	General APL consulting: Migration and Downsizing; Performance Tuning.
Mackay Kinloch Ltd	Consultancy	170-320	Design, analysis and programming for banking, insurance, financial planning and modelling, corporate performance and legal reporting
MasterWork Software	Consultancy	poa	Consultancy
MicroAPL	Consultancy	poa	Technical & applications consultancy.
Ellis Morgan	Consultancy	250-500	Business Forecasting & APL Systems.
Oasis Systems	Consultancy	poa	Expertise in APL system design, Project management, conversion, migration, tuning; for all APL versions (10+ years experience)
Optima	Consultancy	poa	A range of consultants with 3-15 yrs APL PC and mf experience.
QB On-Line	Consultancy	350	Specialising in Banking, Financial & Planning Systems.
RE Time Tracker Oy	Consultancy	poa	Specialised in comprehensive APL Windows user interfaces, APL Multimedia, APL to API level interfacing for Windows, Windows applications, DLLs & databases.
Rex Swain	Consultancy	poa	Independent consultant, 20 years experience. Custom software development & training, PC and/or mainframe.
Rochester Group	Consultancy	poa	Specialise in MIS using Sharp APL.
Snake Island Research Inc	Consultancy	poa	APL interpreter and compiler enhancements, intrinsic functions, performance consulting. APL parallel compiler APEX is giving very good initial performance tests with convolution somewhat faster than FORTRAN.
Strand Software	Consultancy	poa	Advice on migrating to and from all flavours of APL and hardware platforms. Full-screen interface implementation, APL utilities, benchmarking, efficiency analysis, actuarial software, system development tools, valuation, pricing and modelling systems.
Sykes Systems Inc	Consultancy	poa	Complete APL services specialising in audit, optimisation and conversion of APL systems. Excellent design skills. All dialects and platforms. 17-23 years experience.
Uniware	Consultancy (Senior)	FF/day 5000	Consultancy from people with at least 8 years APL experience.
	Consultancy (Senior)	FF/day 7500	Advice and training in Windows programming with APL*PLUS II
	Training	FF10000	5-day class on Windows programming with PLUS II version 4.0
Wickliffe Computer	Consultancy	poa	System design, consultancy, programming and documentation. Especially project management and decision support systems

OTHER PRODUCTS

COMPANY	PRODUCT	PRICES(£)	DETAILS
Adfee	Employment	poa	Contractors and permanent employees
APL People	Employment Agency	poa	Employees placed at all levels.
Bloomsbury Software	Training	poa	Contact the company for details.
ComLog	Comic-Logger	$25.95+p&p	APL*PLUS II comic-book inventory system. Shareware version available on America OnLine.
HMW	Employment	poa	Contractors and permanent employees placed.
HRH Systems	APL lessons		On-screen interactive APL lessons for APL*PLUS, TryAPL2, Sharp and I-APL — in English or French.
	The BBS\APL:	$24 p.a.	703-528-7617, 1200-14400b, N-8-1, 24 hours. APL educational material is downloadable free. An additional 30 megs of APL software for APL*PLUS, PLUS II, IBM, Sharp & I-APL is available to subscribers (cost is $24/yr). Selection available on disk for $15 post-paid. Free on-disk catalogue.
I-APL Ltd	An APL Tutorial	3	45pp by Alvord & Thomson
	An Encyclopaedia of APL (2d Ed)	6	228pp by Helzer
	APL in Social Studies	3	36pp by Traberman
	I-APL Instruction Manual (2d Ed)	3	55pp by Camacho & Ziemann
	APL Programs for the Mathematics Classroom (Springer-Verlag)	16	185pp by Thomson
	Programming in J	10	75pp by Ken Iverson
	Arithmetic	12	118pp by Ken Iverson
	Tangible math	8	36pp by Ken Iverson
	Sharp APL Reference Manual	42	349pp by Berry
	APL Press Books	poa	A comprehensive selection of early APL literature
	Please note there is a packing charge of £3 per order		
Kestrel	Employment	poa	Permanent and contract, home and abroad. From individual placement to supply of complete project teams.
	Software Library	poa	Low-cost software distribution service; call for details.
Oasis Systems	Training	poa	Introductory courses in APL Advanced courses for different APL versions
Renaissance Data Systems	Booksellers		The widest range of APL books available anywhere. See Vector advertisements.
Soliton Associates	MVSLINK	poa	Interface from Sharp APL (Unix & MVS) to non-APL data and software in the MVS environment.
	SSQL	poa	High-performance DB2 interface for Sharp APL (Unix and MVS).

OVERSEAS ASSOCIATIONS

GROUP	LOCATION	JOURNAL	OTHER SERVICES	Ann.Sub.
ACM/SIGAPL	International	Quote Quad		
APL Bay Area	USA N. California	APLBUG	Monthly Meetings (2nd Monday)	$15
APL Club Austria	Austria	-	Quarterly Meetings	200AS(indiv), 1000AS(corp)
APL Club Germany	Germany	APL Journal	Semi-annual meetings	DM60
APL Interest Group	South Africa	-	-	
Ass. Francophone pour la promotion d'APL	France	Les Nouvelles d'APL		
BACUS	Belgium	APL-CAM	Conferences & Seminars	£18 ($30)
Chicago SIG	Chicago			
Hartford Group	Hartford, Connecticut, USA			
Capital PCUG	Washington, D.C.	Monitor	Monthly meetings, occasional classes	free

Danish SIG	Denmark			
Dutch APL Assoc.	Holland		Mini-congress, APL ShareWare Initiative	
FinnAPL	Helsinki, Finland	FinnAPL Newsletter	Seminars on APL	100FIM(private), 30(student), 1000 (Co)
Japan APL Assoc.	Tokyo, Japan			
Melbourne APLUG	Melbourne, Australia		Quarterly meetings	free
New York SIG	New York, USA			
Potomac SIG	Washington DC, USA		Free monthly meetings	
Rochester APL	Rochester, New York			
Rome/Italy SIG	Roma, Ialy			
SE APL Users Grp	Atlanta, Georgia	SEAPL Newsletter		
SOCAL	Southern California		Seminars	$15 ($5 students)
SovAPL	Obninsk, Russia			
SwedAPL	Sweden	SwedAPL Nytt	Semi-annual meetings, seminars	SEK 75
SWAPL	Texas, USA	SWAPL		$18
Swiss APL (SAUG)	Bern	Part of Qtty SI-Info		SF60 (SI) + SF20 (SAUG)
Sydney APLUG	Sydney, Australia	Epsilon	Monthly Meetings	
Toronto SIG	Toronto, Canada	Gimme Arrays!	Monthly Meetings, APL skills database, J SIG, Toronto Toolkit	$25

VENDOR ADDRESSES

COMPANY	CONTACT	ADDRESS, TELEPHONE, FAX, EMAIL etc.
ACM/SIGAPL	Donna Baglio	ACM, 1515 Broadway, New York, NY 10036 USA Tel:+1 (212) 626-0606 Email: baglio@acm.org
Active Workspace Ltd	Ross D Ranson	Moulsham Mill Centre, Parkway, Chelmsford, Essex, CM2 7PX, UK. Tel: 01245-496647; Fax: 01245-496646.
Adaptable Systems	Lois & Richard Hill	49 First Street, Black Rock 3193, Australia. Tel: +61 3 589 5578 Fax: +61 3 589 3220
Adaytum Systems		13 Great George Street, BRISTOL BS1 5RR UK Tel: 0117-921 5555
Adlee	Bernard Smoor	Dorpsstraat 50, 4128 BZ Lexmond, Netherlands. Tel +31-3474-2337, Fax: +31-3474-2342
Andrews	Dr Anne D Wilson	23, The Green, Acomb, YORK YO2 5LL, UK Tel: 01904-792670
APL-385	Adrian Smith	Brook House, Gilling East, York YO6 4JJ UK. Tel: 01439-788385 Fax: 01439-788194 Email: 100331.644@compuserve.com
APL Bay Area Users Group APLBUG	Lewis H. Robinson (Sec)	1100 Gough St, Apt 14A, San Francisco, CA 94109, USA Tel: +1 (415) 928-2058 Email: frgp21a@prodigy.com
APL Club Austria	Erich Gall	IBM Österreich, Obere Donaustrasse 95, A-1020 Wien, Austria
APL Club Germany	Dieter Lattermann	Rheinstrasse 23, D-69190 Walldorf, Germany. Tel: +49 6227-63469 Compuserve: 100332,1461
The APL Group Inc	Stuart Sawabini	644 Danbury Road, WILTON, CT 06897 USA. Tel: +1 (203) 762-3933 Fax: +1 (203) 762-2108
APL Interest Group, South Africa	Mike Montgomery	Private Bag X11, Rivonia 2128, South Africa Tel: +27 (11) 803-7200 Fax: +27 (11) 803-9134 Email: mikemont@spl.co.za
APL People / Software	Jill Moss	The Old Malthouse, Clarence St, BATH, BA1 5NS UK. Tel: 01225-462602
Association Francophone pour la promotion d'APL	Dr. Gérard Langlet	SCM, C.E. Saclay, F-91191-Gif sur Yvette, France. Fax:+33 1 69-08-79-63
Atlantis Software	Arthur Whitney	1105 Harker Avenue, Palo Alto, CA 94301 USA
BACUS	Joseph de Kerf	Rooinberg 72, B-2570 Duffel, Belgium. Tel: +32 15 31 47 24
Beautiful Systems, Inc.	Jim Goff	308 Old York Road, Suite 5, Jenkintown, PA 19046, USA Tel: +1 (215) 886-2636; Fax: +1 (215) 886-4888

The Bloomsbury Software Co Ltd	Peter Day	3-6 Alfred Place, Bloomsbury, London WC1E 7EB UK. Tel: 0171-436 9481; Fax: 0171-436 0524; CompuServe: 100010,1467
Camacho	Anthony Camacho	11 Auburn Road, Redland, Bristol BS6 6LS UK. Tel: 0117-9730036. email: acamacho@cix.compulink.co.uk Reuternet (Sharp): ACAM
Ray Cannon		21 Woodbridge Rd, Blackwater, Camberley, Surrey GU17 0BS UK Tel: 01252-874697 Email: 100430.740@compuserve.com
Paul Chapman		51B Lambs Conduit Street, London WC1N 3NB UK. Tel: 0171-404 5401. Compuserve: 100343,3210
Causeway Graphical Systems Ltd	Adrian Smith, Duncan Pearson	5 The Maltings, Castlegate, MALTON, North Yorks YO17 0DP UK Tel: 01653-696760 Fax: 01653-697719 Compuserve: 100265,1564
Chicago SIG	Larry Mysz	836 Highland Drive, Chicago Heights, IL 60411 USA C'serve:73040,3032
Cinerea AB	Rolf Kornemark	Skyttegatan 25, S-193 00 Sigtuna, Sweden.
CODEWORK	Mauro Guazzo	Corso Cairoli 32, 10123 Torino, Italy. Tel: +39 11 885168 Fax: +39 11 812 2652
ComLog Software	Jeff Pedneau	PO Box 5570, Derwood, MD 20855 USA Tel: +1 (301) 990-7063 Email: jeff@softmed.com
CPCUG	Lynne Startz	Capital PC User Group, 51 Monroe Street, Suite PE-2, Rockville, Maryland 20850-2421, USA. Tel: +1 (301) 762-9372 Fax: (301) 762-9375.
David Crossley		187 Le Tour du Pont, Quartier Le Mourre, 84210 ST DIDIER, France Tel: +33 90-66-08-87
CYBEX AB	Lars Wentzel	Gruvgatan 35B, S-421 30 V. Frölunda, Sweden. Tel: +46 31-45 37 40. Fax: +46 31-45 24 23.
Peter Cyriax Systems	Peter Cyriax	22 Hereford Road, London W2 4AA UK. Tel: 0171-229 5344
Danish User Group	Per Gjerlof	Email: gjerper@inet.uni-c.dk
Datatrade Ltd.	Ian Tomlin	1 & 2 Sterling Business Park, Salthouse Road, Brackmills, Northampton, NN4 0EX UK. Tel: 01604-760241
Dogon Research	Dick Bowman	2 Dean Gardens, London E17 3QP UK Tel: 0181-520 6334 Email:bowman@apl.demon.co.uk
Dutch APL Association	Bernard Smoor (Sec)	Postbus 1341, 3430BH Nieuwegein, Netherlands. Tel: +31 3474-2337
Dyadic Systems Ltd.	Peter Donnelly	Riverside View, Basing Road, Old Basing, Basingstoke, Hants RG24 0AL UK. Tel: 01256-811125 Fax: 01256-811130
E & S Associates	Frank Evans	19 Homesdale Road, Orpington, Kent BR5 1JS UK. Tel: 01689-824741
Evestic AB	Olle Evero	Berteliusvagen 12A, S-146 38 Tullinge, Sweden Tel&Fax: +46 778 4410
FinnAPL		Suomen APL-Yhdistys RY, FinnAPL RF, PL 1005, 00101 Helsinki 10, Finland
General Software Ltd	M.E. Martin	22 Russell Road, Northholt, Middx, UB5 4QS UK. Tel: 0181-864 9537
Greymantle Associates	George MacLeod	Bartrum House, Ravens Lane, Berkhamsted, Herts, HP24 2DY UK Tel: 01442-878065 Email: 100412,1305@compuserve.com
Hartford CT Group	Bob Pomeroy	Mass Mutual Life, 1295 State St, Maildrop F465, Springfield, MA 01111 USA Tel: +1 (413) 788-8411x2838
H.M.W.Trading Systems Ltd	Stan Wilkinson	Hamilton House, 1 Temple Avenue, Victoria Embankment, London EC4Y 0HA UK. Tel: 0171-353 8900; Fax: 0171-353 3325; Email:100020.2632@ compuserve.com
HRH Systems	Dick Holt	3802 N Richmond St, Suite 271, Arlington, VA 22207 USA Tel: +1 (703) 528-7624; Email: dick.holt@acm.org
Michael Hughes		28 Rushton Road, Wilbarston, Market Harborough, Leics. LE16 8QL UK. Tel: 01536-770998
IAC/Human Interfaces	Ian A. Clark	9 Hill End, Frosterley, Bishop Auckland, Co. Durham DL13 2SX UK Tel: 01388-527190. Compuserve: 100021,3073
I-APL Ltd	Anthony Camacho (for queries, order forms)	11 Auburn Road, Redland, Bristol BS6 6LS UK. Tel: 0117-9760036 email: acamacho@cix.compulink.co.uk Reuternet (Sharp): ACAM
	J C Business Services (for pre-paid orders only)	56 The Crescent, Milton, Weston-super-Mare, Avon, BS22 8DU UK Tel: 01934-625181
IBM APL Products	Nancy Wheeler	APL Products, IBM Santa Teresa, Dept M46/D12, 555 Bailey Avenue, San Jose CA 95141, USA. Tel: +1 (408) 463-APL2 (=2752) Fax: +1 (408) 463-4488 Email: APL2@vnet.ibm.com Cserve: GO IBMAPL2
Impetus Ltd	Cedric Heddle	Rusper, Sandy Lane, Ivy Hatch, SEVENOAKS, Kent TN15 0PD UK Tel: 01732-885126

INFOSTROY	Alexei Miroshnikov	3 S. Tulenin Lane, St. Petersburg 191186 Russia. Tel:+7 812-312-2673 Fax:+7 812-311-2184 Email:aim@infostroy.spb.su
Insight Systems ApS	Morten Kromberg	Nordre Strandvej 119A, DK-3150 Hellebæk, Denmark Tel:+45 42 10 70 22 Fax: +45 42 10 75 74 Email: insight@inet.uni-c.dk
Intelligent Programs Ltd	Mike Bucknall	9 Gun Wharf, 130 Wapping High St, London E1 9NH Tel: 0171-265 1120
Interprocess Systems Inc.	Stella Chamberlain	11660 Alpharetta Highway, Suite 455, Roswell, Georgia 30076, USA Tel: +1 (404) 410-1700. Fax: +1 (404) 410-1773 Cserve: 70373,2676
Iverson Software Inc.	Eric Iverson	33 Major Street, Toronto, Ontario, Canada M5S 2K9 Tel: +1 (416) 925-6096; Fax: +1 (416) 488-7559
JAD Software	David Crossley	580 Eyer Drive, #81 Pickering, Ontario, Canada L1W 3B7 Tel: +1 (905) 837-1895 Fax: +1 (905) 831-5172
Japan APL Association		23-2-302 Hiromichi, Adachi-ku, Tokyo 120, Japan
Kestrel Consulting	Mark Harris	Business & Technology Centre, Bessemer Drive, Stevenage, Herts SG1 2DX UK. Tel: 01438-310155 Fax: 01438-310131
Lingo Allegro USA Inc.	Victoria H. Fil	113 McHenry Road, Suite 161, Buffalo Grove, IL 60089 USA Tel:+1 (708) 459-7529 Fax:+1 (708) 459-8501 Email: info@lingo.com
Mackay Kinloch Ltd	Alastair Kinloch	519 Webster's Land, Edinburgh EH1 2RX, Scotland, UK Tel/Fax/Answerphone: 0131 228 3580 Pager/Voicemail: 01426 98 3858 Compuserve: 100010,33
MasterWork Software Ltd	Fraser Jackson	PO Box 56-036, Tawa, Wellington, New Zealand. Tel and Fax: +64 (4) 232-4440 Email: 100242.2535@compuserve.com
Melbourne APL Group	Harvey Davies	CSIRO Div Atm Res, Private Bag No.1, Mordialloc, Victoria 3195, Australia Tel: +61 3 586 7574 Fax: +61 3 586 7600 Email: hld@dar.csiro.au
Mercia Software Ltd.	Gareth Brentnall	Holt Court North, Heneage Street West, Aston Science Park, Birmingham B7 4AX UK. Tel: 0121-359 5096. Fax: 0121-359 0375
MicroAPL Ltd.	David Eastwood	South Bank Technopark, 90 London Road, LONDON SE1 6LN UK Tel: 0171-922 8866 Fax: 0171-928 1006 Email: MicroAPL@microapl.demon.co.uk
Ellis Morgan	Ellis Morgan	Myrtle Farm, Winchester Road, Stroud, Petersfield, Hants UK. Tel: 01730-263843
New York SIG APL	Nestor Nelson	PO Box 138, NY 10185-0002, USA
Oasis Systems	Theo Zwart	Lekstraat 4, 3433 ZB Nieuwegein, Holland Tel: +31 30 60 66 336 Fax: +31 30 60 65 844 Email: 100447.431@compuserve.com
Optima Systems Ltd	Paul Grosvenor	Airport House, Purley Way, Croydon, Surrey CR0 0XY UK. Tel: 0181-781 1812 Fax: 0181-781 1999
Potomac APL SIG	John Martin	51 Monroe Street, Plaza East Two, Rockville MA 20850-2421 USA Tel: +1 (301) 762-9372 Fax: +1 (301) 762-9375 Email:jam@acm.org
QB On-Line Systems	Philip Bulmer	5 Surrey House, Portsmouth Rd., Camberley, Surrey, GU15 1LB UK. Tel: 01276-855880 Fax: 01276-855301
Renaissance Data Systems	Ed Shaw	PO Box 421, Georgetown, CT 06982, USA. Tel: +1 (212) 864-3078
RE Time Tracker Oy	Richard Eller	PO Box 363, FIN-00101 Helsinki, Finland. Tel: +358-0-400 2777
Rochester APL	Gary Dennis	Soliton Associates, 1100 University Avenue, Rochester, NY 14607 USA Email: gsd@ipsalab.tor.soliton.com
The Rochester Group Inc.	Robert Pullman	50 S.Union St., Rochester NY 14607-1828, USA. Tel: +1 (716) 454-4360. Fax: +1 (716) 454-5430
Rome/Italy SIG	Mario Sacco	Casella Postale 14343, 00100-Roma Trullo, Italy Email: marsac@vnet.ibm.com
SE APL Users Group	John Manges	991 Creekdale Drive, Clarkston, GA30021 USA
Shandell Systems Ltd.	Maurice Shanahan	Chiltern House, High Street, Chalfont St. Giles, Bucks HP8 4QH UK.
Snake Island Research Inc	Bob Bernecky	18 Fifth Street, Ward's Island, Toronto, Ontario M5J 2B9 Canada Tel: +1 (416) 203-0854 Fax: +1 (416) 203-6999 Email: bernecky@eecg.toronto.edu
SOCAL (South California)	Roy Sykes Jr	Sykes Systems Inc, 4649 Willens Ave, Woodland Hills, CA 91364-3812 USA Tel: +1 (818) 222-2759 Fax: +1 (818) 222-9250
Soliton Associates	Laurie Howard	Soliton Associates Ltd, Groot Blankenberg 53, 1082 AC Amsterdam, Netherlands Tel: +31 20 646 4475 Fax: +31 20 644 1206 Email:sales@soliton.com
SovAPL	Alexander Skomorokhov	PO Box 5061, Obninsk-5, Kaluga Region, Russia Email:askom@apl2.obninsk.su

Strand Software Inc	Anne Faust	19235 Covington Court, Shorewood MN 55331 USA Tel: +1 (612) 470-7345 Email: amfaust@aol.com
Rex Swain	Rex Swain	8 South Street, Washington, CT 06793 USA. Tel: +1 (860) 868-0131 Fax: +1 (860) 868-9970 Email: rhswain@pcnet.com
SWAPL	Stuart Yarus	PO Box 210367, Bedford, Texas 76095, USA Tel: +1 (817) 577-0165 Compuserve: 73700,2545 Email:syarus@unicomp.net
SwedAPL	Christer Ulfhielm	Novator Consulting Group AB, Svärdvägen 11C, S-182 33 Danderyd Sweden Tel: +46 8 622 63 50 Fax: +46 8 622 63 51 CServe: 100341,404
Swiss APL User Group		Swiss APL User Group, CH-3001, Bern 1, Switzerland Email: si@ifi.unizh.ch
Sydney APLUG	Rob Hodgkinson	PO Box 1511, Macquarie Centre, NSW 2113, Australia Tel:+61 2 257 5313
Sykes Systems Inc	Roy Sykes Jr	4649 Willens Ave., Woodland Hills, CA 91364, USA Tel: +1 (818) 222-2759 Fax: +1 (818) 222-9250
Toronto SIG	Ben Best	PO Box 55, Adelaide St. Post Office, Toronto Ontario M5C 2J5, Canada Email: benbest@io.org http://www.sigapl.mtnlake.com/sigapl/welcome.html
Uniware	Eric Lescasse	Tour Neptune, Cedex 20, 92086 Paris la Defense 1, France. Tel: +33 (1) 47-78-78-00. Fax: +33 (1) 40-90-04-11
Wickliffe Computer Ltd	Nick Telfer	76 Victoria Rd, Whitehaven, Cumbria, CA28 6JD UK. Tel: 01946-692588
Warwick University	Prof. Jeff Harrison	Dept of Statistics, University of Warwick, Coventry, CV4 7AL UK Tel: 01203-523369
Zark Incorporated	Gary A. Bergquist	23 Ketchbrook Lane, Ellington CT 06029, USA. Tel: +1 (203) 872-7806

WORLD WIDE WEB SITES (checked Jan 1996)

ORGANISATION	URL
IBM APL2 homepage	http://www.torolab.ibm.com/ap/apl/apl2.html
Mackay Kinloch Ltd	http://ourworld.compuserve.com/homepages/Alastair_Kinloch
SigAPL	http://www.acm.org/sigapl/
Rex Swain	http://www.pcnet.com/~rhswain
Toronto SIG	http://www.sigapl.mtnlake.com/sigapl/welcome.html

FTP SITES (checked Jan 1996)

ORGANISATION	DOMAIN NAME
IBM APL2	ps.boulder.ibm.com/ps/products/apl2/
Toronto toolkit	see Toronto SIG home page
Waterloo Archive	archive.uwaterloo.ca/ftparch/languages/apl

Experiences with the Use of Causeway

by Jan Karman (a naïve user)

Introduction and Orientation

A causeway (according to the dictionary) is a paved way (derived from the Latin 'calciata' meaning as much as 'chalked' or 'paved with lime stone'). In modern times, I assume, it is a highway or turnpike with fly-overs and rest areas. I am a naïve user because I am not a professional computer scientist but just somebody who needs software for his (or somebody else's) applications. I showed very naïve indeed when I first started with Causeway in a rather large and complex application: a pension fund administration – not as a free choice, it just fell on my route like manna.

I didn't know Causeway (who did outside Swansea, UK?), except from reading Adrian's article in *Vector* and later Duncan's. I just trusted the promises and accepted the assignment.

I know most ins and outs of pension funds, have a few years' experience with Dyalog APL/W, but knew little of the internals of Microsoft Windows (I'm just a 'clicker'). I bought myself the recommended book *'The Windows Interface, An Application Design Guide'* and started.....

Why did I trust the promises? Adrian Smith is one of my first APL teachers (although perhaps he doesn't know so). His book *'APL, A Commercial Design Handbook'* is on my shelf (it wasn't for a long time, since it was on my desk), together with *'APL, the Language and its Usage'* (Polivka, Pakin) and *'APL, An Interactive Approach'*(Rose, Gilman).

First, I needed to adjust my home-made character-based database management system (because there is so much functionality specific to the kind of applications and so little overhead that I prefer this above a manufactured one).

Here already the approach of the Causeway-design paid off (that's what I mean by the presence of rest areas: it pays off to carefully look around and to look inside). I re-designed my dbms in a structure similar to the design of Causeway: fully table-driven, using the TextMatrix Grid object, and stored all the functionality in a filetab (similar to classtab, styletab, domain, etc.), containing file and fields characteristics, constraints, display and storage formats, and so on, and even code as much as I want 'outside' the workspace; it has 12 columns now.

Browsing through the Causeway system I came across another word for which I needed the dictionary (remember I'm Dutch): 'holler'. What does 'holler' mean? The description in the dictionary and the functionality which I found out made me think of a long forgotten public service which I remembered from the (in those days) remote and isolated little village in the SW of Holland where I was born: the bell-man. Loudly and clearly make everybody know the important changes in life!

Appearance

If one is going to use Causeway seriously I would recommend to install the Causeway font. (Guidelines come with the product.)

Warning: if you are also a user of Rex Swain's *WSFIND* this function fails under the Causeway font (how can this happen?) and you'll need to modify this function. The Causeway font is a piece of beauty (of the already beautiful design of the APL character set itself) and certainly makes APL more approachable, due to the locations of the characters on the keyboard (e.g. <commute> with Shift+Ctrl+t).

Customizing

Naïve I showed when I thought I needed a multi-line TextMatrix Grid: I had to adjust the *NavGrid* function (there is very instructive code in there!), and thus, I had to dive in the deep right away. I learned a lot about Causeway, how it was built and how to customize it to my own needs. I came across the DT-domain, which had been copied from Microsoft Excel, on its turn copied from Multi Plan, and discovered that the year 1900 is a leap-year! (Is this why Dyalog's dates stop backwards at March 1,1600?. Keep away from leap-years!). Result: all your dates display one day short, if you did a correct conversion (e.g. from old data), that is. Pension funds have a long range of dates and periods: from say 1890 until far in the next century, and many ways of treatment of dates. Thus, you better write and add your own (dates) domains as needed. It was stated in the guidelines that Domain had not been generalized.

Designer

The Dbx-designer is a miracle. It works! and it works efficiently, at least and certainly for a rough design. Refinements can be made afterwards easily by changing values in the 'dbx_xxxx' variable by use of enclose and pick, e.g.

```
    ((1 4)⊃dbx_xxxx) ← ⊂ new value
```

in order to change the size of the main Form, or

```
    ((1 2)(4 9)⊃dbx_xxxx) ← ⊂ new value
```

in order to change the call-back function of the first event of the fourth object, or

```
    ((1 4)(4 9)⊃dbx_xxxx) ← ⊂ new condition
```

and so on. (I do not understand why the order of the columns shown in the designer is different from those in the dbx-variable.)

Use in Practice

The ease of making and positioning all objects and decorations: designing the main menu, designing menus for the right mouse button, messages in the StatusBar, ProgressBars, etc. is amazing. There is a rich choice of objects, styles and prefab-ed events.

Cloning, aligning, the Snap of 1, 4, 8 are all tools which make life comfortable.

Then, as soon as you've designed some of these dbx's, you may choose lines from existing and copy them into new ones, or use several lines as a 'template', i.e. often the first few lines. For the application I built there is a certain pattern: most dbx's have an identification (of the participant) section, a combo selector (for selecting by name), a data section, an annotation, a control section (action buttons) and a status bar (sometimes also a progress bar). This 'geography' shows that some of the sections are shared by most dbx's.

Further, you might write a function to 'sort' the dbx: take care that the FM (Form) is on top, next is the Combo Selector and the remaining ones are sorted by group and/or position. The result is such, that the user always has the first focus at the Combo Selector and that sequential focusing by Alt+Tab is always in the order of the relevant objects. I also wrote a function to discard redundant classes from the classtab: this often saves a lot of space in your workspace.

Performance

The sting is in the tail. Once you have designed the dbx's they let you wait! Performance, that is, emerging of the windows on your screen, needs more than 'a touch more' compared to directly designed dbx's. (They behave well on a Pentium 75Mhz and up.)

A few hints:

- be moderate in the number of objects in each dbx (max. 15 to 20).
- use 'Hide' and 'Show' commands in a particular session. Once the dbx has been created it's much faster to make it visible and unvisible.
- try to be efficient in 'hollering' variables and 'refreshing' objects, avoid redundant execution of (call-back) functions.

Support

The Help is omni-present.

The support (which you definitely will need: after three months working with Causeway I still find little spots and stains) is informal but professional, fast and to the point, completely in line with the style of support you are used to getting from Dyadic. They (Adrian and Duncan) manage to give support with a personal 'touch' and in relation to the 'contribution' (what else is $50 for such a sophisticated product?) of high quality and – as they put it in their ad – provide a 'shoulder to lean on when times get hard'.

Conclusion

My conclusion is that Causeway is a high-sophisticated product which only has its equals in APL (or J) itself or in Dyadic's APL interpreter, and which could lead to a renaissance of APL in many regions. At least the customers are surprised and like the results!

Managing Objects in Causeway

from Jon Sandles

Introduction

Managing one's code is something that often comes last in the list of priorities for the lone system developer. When working in teams, certain standards are often laid down to help the team develop together. In the modern Windows development environment (which is object-orientated by nature) object management tools are available in a number of different forms. These are often the most expensive parts of a toolkit — particularly for multi-user environments.

In this article I take a look at some of the problems I have run into in using Causeway for Dyalog APL, which has no built-in object management. In a future *Vector* I will be reviewing JAD SMS which is an object management toolkit for APL*PLUS II v5. We hope to review other such systems (including non-APL systems) in future issues of *Vector*. This will end in a proposal for what functionality is required to manage objects effectively in an APL Windows environment.

Upgrading Causeway

Causeway is a method of defining Windows applications that can be implemented on any APL platform. The advantages of using this generic and portable platform are many — and of course there are also disadvantages. I am going to talk about one of the disadvantages of Causeway — upgradeability.

The features I am referring to here have occurred in upgrading Causeway releases within Nestlé in versions of DyalogAPL up to version 7.1.2. Because Causeway is not part of Dyalog APL, but an extra layer of APL code which interprets your Window definitions, you often have to upgrade the two together or at different times. One of the great advantages of Causeway — that it allows the user to hide mistakes in the Gui of the interpreter and fix them with workarounds — is also its downfall — when the interpreter writers fix the problem the Causeway code often needs readjusting.

Normally when faced with upgrading just an APL interpreter I would hope that the job is reasonably painless: take backups of everything — install the new version — check all my workspaces still work — if they do forget all about the previous version and see if there's anything new that's worth using. Sometimes the workspace format changes (but the old format still loads) and newly saved workspaces must be loaded in the new interpreter — in a multi-user pc environment this may mean that careful synchronisation of the upgrade is required across a number of pcs/network servers. But as long as this is done with reasonable care then problems are normally few and far between as long as the interpreter itself has been thoroughly tested.

Causeway itself is written in DyalogAPL. It resides in, and is called within, every workspace that requires it. It is not ideal (as with any utility code) to have different versions within different workspaces. Thus, when you get a new release of Causeway you will probably want to test it in a number of environments, and then when you are happy you will want to install the new version (for new developments) but also upgrade existing code onto the new version — to prevent confusion — to help support and potentially to ease existing bugs.

Unfortunately Causeway exists inside and outside your workspace in a number of ways — this makes the upgrade path treacherous at best. The first step is obviously to upgrade the Causeway code. This is pretty simple, something like `)copy causeway` should do the job and replace the old Causeway code with the new. With any luck the Causeway developers will have documented what has changed and there may even be some functions that aren't used any more — `)erase` these if you are feeling really brave.

In a simple world the above would be the end of the story. You would test the workspace out, find it still works OK and go home happy. Unfortunately, other changes are often required.

The Causeway form-definitions are held in nested variables within your workspace. Due to progress, design changes etc. (remember the current Causeway architecture is fairly new) these definitions are often adjusted, meaning that every form definition requires upgrading. Hopefully you have used some standard naming convention so that it's easy for you to find all your form definitions. Also with any luck the Causeway developers have supplied the code to upgrade each form definition automatically. If this is true it should be easy to convert them all.

Sometimes, there are changes within Causeway that require that form definitions are analysed to see if they use defunct procedures/concepts. This will probably have to be done manually — although I would advise you write a little routine that searches the forms for any such 'problems' and tells you which ones need attention. These sorts of changes are quite common. Recently a Causeway event was removed from all the supplied objects — the PC/Post Create event. This event could be used to run code following the creation of the object. Thus any code within a form relying on this event had to be modified. In this example this code could usually be moved onto the CR (create event) or moved onto a different object.

More advanced users of Causeway may have also modified or extended their Causeway environment in a number of ways. They may have added styles, they may have added objects, they may have added global events, they may have added column types for the grid object etc. If changes like this have been made, then much greater care is required — following the above path when upgrading will end in tears.

Adding/Changing Styles and Objects in Causeway

Causeway maintains a list of styles in the global variable `∆styletab`. Any modifications you make to this table will be lost when you copy in the new Causeway code — thus you should always take a backup copy and re-apply any changes you make to the new version.

Causeway objects are held in a large global nested variable called `∆classtab`. Once again modifications will be lost when the new code is copied in. Fortunately classes can also be held externally to the workspace in APL sharefiles. The best path to upgrading the class table is not to take a copy of the class table but to make a note of which classes it contains before copying the new Causeway workspace in, and then rebuild the class table from scratch from the external file copies of the classes.

Unfortunately, in the last few Causeway releases, changes have been required to classes when upgrading from one version to another. Causeway comes with a standard set of classes so these should all work — you can then generate the file copies of the classes from these. For those classes that you have created/modified you will need to redo any modifications by hand and then save the classes to file. The new set of class files can then be used to regenerate all the class tables in all the upgraded workspaces. Because of this I advise that you track quite carefully when you are using standard Causeway objects and when you modify/create your own. This will enable you to perform the upgrade in a much more controlled manner.

Adding/Changing Events in Causeway

Once again the set of events that causeway objects handle and monitor is held in another global variable ... `∆events`

If you added/modified these then you should take a backup copy of this variable before you upgrade and then update the new copy with any of your changes. This is rarely necessary now because Causeway classes now support internal/local events which is why you often had to add events to the list to support particular objects (e.g. grid-specific events). This type of event is now held within the object definition itself and not globally in the events table.

For the Future

Causeway has been designed in such a way that it is an 'open' product. This encourages modification and creation of your own objects. Indeed, this openness is one of the strengths of the product. But when upgrading, these user changes are often lost, or cause errors which may well be difficult to debug or fix without advanced knowledge of the architecture. The root cause of this is the use of large nested global variables to hold the extensible parts of the Causeway architecture. This means that when upgrading, you need to re-hook your extensions into these core variables — often having to check for architectural changes along the way — and then perform upgrades from your extended versions. Also, being a developing product it is still at a stage when concepts are added/removed — meaning that if you upgrade you must re-adjust your application code. This aspect should not be a problem as the product stabilises.

The Causeway architecture forms the core of all Nestlé's DyalogAPL systems. Because support and development analysts are often the same people, it is too confusing and dangerous to allow different versions of the architecture to co-exist in a production systems environment — thus whenever the Causeway architecture changes many systems need to be re-adapted, so upgrading needs to be as seamless as possible. At the moment it takes too long and is too complex a task.

The basic Causeway architecture is only partially the cause of these problems. People who have been using Causeway would not have typically run into these difficulties unless they have been customising the system themselves. This customising is what we in Nestlé see as being the real strength of the system, but unfortunately it is also a disadvantage when there is no effective object management environment. With an object management environment in place upgrades could be handled far more transparently, older versions of objects would be kept — meaning upgrades could be made where appropriate/ convenient — and different versions of the Causeway code could be maintained in parallel. So what is it that makes an effective object management tool? Future *Vectors* will hopefully tell us ....

KPS: Beyond the Spreadsheet

by Ian A. Clark

KPS stands for "Kunzle Planning System", the brainchild of George Kunzle and the flagship product of Bristol-based Adaytum KPS Software, a small but growing company of currently under 30 employees. In mid-1995 Adaytum acquired a 100% holding in Insight Systems, a Danish software house well known in APL circles. As a result of the impetus this gave to its development plans, Adaytum was able to announce KPS for Windows at the SoftWorld Exhibition, Birmingham, England, in October, 1995, where it demonstrated a working prototype in a blaze of publicity.

At the time of writing, first customer shipment of the Windows product is still a week or two away, but the wraps are off and the shape of the product is there for all to see. What is less common knowledge is that Adaytum has been selling KPS as a DOS product successfully enough to turn it from a tiny startup into a £2M turnover company in the space of three years. In the UK alone there are now 200 KPS customer sites. KPS is almost entirely written in (Dyalog) APL, so it can already lay claim to being one of the highest earning APL applications ever written. But of course the customer couldn't care whether it is written in APL, C, C++ or Esperanto. It looks (superficially) like any other Windows integrated spreadsheet package. (The development team happens to think that's an enormous achievement.)

Why has it been so successful? KPS is a Business Management System designed to overcome the limitations of the spreadsheet when used for business planning and budgeting. Now a spreadsheet is a pretty blunt instrument, in spite of years of refinement from the original humble VisiCalc on the Apple II and the Commodore Pet to the dizzy heights of Excel and Lotus 1-2-3. It's designed to provide quick solutions to simple problems. A complex business management system importing, consolidating and sharing substantial data across an enterprise is somewhat beyond their remit.

Yet this is how many enterprises use them, and hate themselves for it. Just how they got themselves painted into this corner awaits the definitive psycho-historical treatise, which long-serving operations research (OR), management information system (MIS) and data processing (DP, ADP, IT) specialists will covertly ponder in shocked silence. But the outcome is no secret. Many organisations rely on a mess of incompatible, inflexible, stand-alone spreadsheet applications; nobody but the original macro-writer knows how they work, and not even they can alter them reliably. Adding new brands, product codes or divisions typically takes weeks of re-programming, in a fairy-language which frankly won't take the strain. (It's rather fun being able to write this in an APL publication.)

So what do customers see in KPS? Quite simply KPS out-performs all state-of-the-art spreadsheets in its chosen area, which Adaytum calls "Budgetware". It does this by offering many small features, and one or two great ones, well-adapted to the needs of financial managers, planners and cost accountants, in medium-to-large organisations. Quoting from its brochure KPS provides the ability to:

- Support a single integrated system across departmental and functional areas.
- Import and consolidate large volumes of data from multiple sources in minutes, not days.
- Build or change complex models quickly and easily without recourse to specialist support.
- Produce ad-hoc reports on demand.
- Perform comprehensive "what-if" and variance analysis by product, region, customer or any other category.
- Allocate overheads according to any criteria.

This last item may sound woolly, but it concentrates in six words one of the most impressive features of KPS, its ability to allocate an altered total across the contributing values, which may themselves be computed from other values.

For example, suppose you have the following row of figures, the end one being a total of the foregoing ones:

```
    100   100   150   200   100   50   300   1000
```

Now overtype the total: 1000, making it, say, 1100. With a spreadsheet set up to do this straightforwardly, you would simply lose your formula and replace it by a constant, 1100. But with KPS (which does it even more straightforwardly) the following happens:

```
    110   110   165   220   110   55   330   1100
```

Altering a total (in this case to increase it by 10%) causes the contributing figures to be altered in the right way (in this case increasing each of them by 10%) to add up to the new figure. If the contributing figures themselves are totals, then their contributors in turn get adjusted, all the way back to what KPS calls "detail" items, i.e. non-calculated. When you realise that what KPS calls a total can have a complex formula in plus, minus, multiply, divide, power, if-then-else and even complex functions like feeds, you realise that this is an enormously powerful facility, with mathematical depths hard to plumb.

The budgeting implications are easy to see, however. And to use. You, the forecaster, can look at a business plan as a multi-dimensional array (a *d-cube*, in KPS parlance), choosing whatever slice you care to portray as the rows and columns of a spreadsheet-like grid, then you can say things like "let's increase this cell by 10% and see what changes". If too many cells change (they change colour as well as contents), you can go back a step and "hold" the cells you don't want to change.

You needn't start with actual typed-in figures, either. There are quick ways of setting the grid to zero or a constant and filling planes of the grid with numbers which grow in a rich variety of ways. Even the allocation facility can be pressed into service to create convincing time-series from scratch. KPS understands times and dates. It can allocate values across a monthly timescale in proportion to the number of days in the month, getting the leap-year right, too. And all without you having to type a row of figures like this: 31 28 (or is it 29?) 31 30 ... (although you can do that too, if you like).

Some people think that using KPS is more like using a painting program than a spreadsheet, with figures pouring into their designated areas like colours. So much so that one major UK brewery could boast "it used to take nineteen people three months to produce one quarterly forecast. It now takes one management accountant half a day". And of course you can generate respectable reports and graphs from the sets of figures when you've produced them, so nobody can accuse you of just having fun.

It's remarkable how quickly you can set up a d-cube from scratch. Nobody likes to talk about hypercubes (nasty frightening things) or even multi-dimensions, if it can be helped. But tables of figures in an organisation tend to fall into a small number of patterns. There may be lists of product codes, or sales regions. There may be typical timescales consisting of days of the week, hours of the day, 5-year periods split up into quarters, or this year and next year split up into months. In addition to the raw-value columns there may be grand totals, totals for the year or half-year, and so forth. KPS lets you build and save each of these sets of column headings as a library entity, called a *d-list* (short for dimension-list). The calculation needed for a given total is attached to the d-list itself, whereas a spreadsheet attaches a calculation to each cell and replicates it (in some more-or-less perplexing way) when you ask it to. Therefore, for each "total" d-list item (aka column-heading or row-heading) there is a single formula, which applies to a whole column (or row) in a 2-dimensional d-cube, a whole plane in a 3-dimensional d-cube, and in general (since we're talking to APLers) a whole (n-1) matrix of cells in the general case of an n-dimension d-cube. Never mind. If you're an accountant, you simply define a new d-cube in terms of a collection of n d-lists and hey presto! — a d-cube appears full of zeros.

You can then feed data in from a variety of sources by marrying up relational domains with d-list items, ASCII file fields, in fact any data source for which there is an ODBC driver (and that's most commercial products nowadays). Nor do you need to confine the "transfer link" as it's called to (one number)—(one cell), but you can "consolidate" (many-to-one) or "allocate" (one-to-many). It's particularly attractive to transfer data between different d-cubes sharing one or more d-lists in common, and quick to set up. In that way a large organisation can pass data up the managerial pyramid, consolidating or allocating data across subheadings at each stage. Activating a transfer link is as simple as pressing a button. KPS allows you to build a "radar map" — a layout of buttons to look like an organisation chart or a flow diagram. Some of the buttons can display given d-cubes, others can activate named transfer links.

This raises the question: "where did that number come from"? There's another button with an electric drill icon. Pressing this will progressively "drill down", opening windows into d-cubes or ASCII file images to show you the contributory numbers.

Curiously, there's interest in KPS among organisations who have problems simply converting data from one form to another between different subsystems. The potential for sophisticated import and export offers a sort of user-friendly patchboard, without needing to pour the data into a d-cube at all. But of course, when you've got the Electric Light, why use it only when you're trimming the candle wicks?

Ian A. Clark
Adaytum Software
13 Great George Street
Bristol BS1 5RR

Recent Meetings

This section of Vector documents all British APL Association meetings, and any other events of interest to the APL world. If you have recently attended any gathering which you feel would be interesting to Vector readers, please let the Editor have a brief note, and we will include it here.

Guide-Share APL Day with APL-Club of Germany Berlin, 12th October 1995

notes by Adrian Smith

Introduction

Firstly, many thanks to Dieter Lattermann for inviting Duncan Pearson & myself as visitors to this joint session of Guide-Share (Europe) and the APL-Club of Germany. It gave us a good insight into the way APL is moving in Germany, as well as the opportunity to meet lots of keen and thoughtful APLers.

I think the most important message we brought back is that OS/2 (and with it APL2/OS2) is solidly established in continental Europe, and that if interpreters like APL+Win and Dyalog/W are to make any headway here, they really need to push forward with their APL2 compatibility. There seems to be a lot of interest in IBM's view of co-operative processing, with several papers discussing real applications built using mainframe DB2 databases with an OS/2 front-end in APL2.

Presenting Election Statistics

Hans-Georg Felinks, City of Dortmund

This was the only paper of the day presented entirely in (idiomatic) German, and I am sorry that I failed to appreciate most of the jokes. The subject matter was most interesting, being a description of an APL2 application for analysing and presenting local election results in Dortmund. The scope of the system was 606 voting districts, with 458,000 people entitled to vote. The system processed the results as they arrived in a central IMS database, aiming for "midnight publication" of ward figures, comparison with eve-of-poll forecasts, demographic trends and so on.

Selection and categorisation was driven by a set of 'mask arrays' generated by the APL2 program, and the results could be tabulated or graphed in a variety of ways, including a colour-shaded map of the Dortmund area.

Apologies for not giving a better account of this one – it would be good to have a full write-up of this work in a future Vector.

Managing Futures and Options with APL2

Alfred Wettach, Allianz Investment Company

This talk described a large administrative ("Back-office") APL2 system which has been operational at Allianz since March 1994. It currently handles Futures and Options — they are working to integrate Swaps. It is probably worth repeating Alfred's definition of what these actually are:

- **Futures** are agreements between two parties to make an exchange at a definite future date. The price is fixed now.
- **Options** give one of the parties to the deal the right to buy or sell a particular good for a fixed price within a given period (USA) or at a given date (EU). You pay a small amount now for the option, and perhaps you may pay the rest in the future.

The key point is that 98% of these deals are never exercised! A company which has a large number of such "derivative assets" on the books clearly has a tough job knowing whether it is in profit or loss at any given moment, as it must make assumptions about the likely uptake of options, and estimate the unrealised asset values accordingly. This is where Nick Leeson came unstuck!

The situation is complicated by legal restrictions, and by the crucial difference between an option to sell (known as a "short" position) and an option to buy (when a trader is "long" instead). If you are "long" you just risk the option price — you do not have to buy; however if you are "short" you must sell if your opposite number wishes to buy. Clearly this can be a much bigger risk if you have offered something you don't actually have.

Putting a Gui Interface on Legacy Systems

Jim Brown (IBM APL2 Development Group)

Jim's talk began with an example of using APL2/OS2 as a subtitute session manager for mainframe applications which used exclusively Quad and Quote-Quad prompting to select from menu choices, and which then output text reports direct to the APL session.

The trick is to use shared variables over AP119 (the TCP/IP processor) to redirect session output to a network socket in place of the mainframe session manager. The mainframe system types a list of options:

```
Please select one of the following ...

      1. Fish and Chips
      2. Sausage and Beans
      3. Pizza
      4. Lasagne
      0. Exit this menu

⎕:
```

... and the user types a number. The program then either goes to another menu, or shows some kind of report. There are three things people typically do with the report: (a) read it, (b) print it, (c) type it into Excel. If instead of a mainframe terminal you have the OS/2 session, you can at least use cut and paste to speed up (c), so here is a useful gain in productivity at zero cost.

Stage two is to field the option list, recognise it as a set of choices and let the user click on a list box instead of typing a number. A little programming is required, but the cost is still near to zero, and suddenly the user has a real Gui application!

Jim continued with a brief outline of the shared-variable approach to Gui objects which IBM first showed at San Antonio. This has an appealing simplicity, but I can't help feeling that it would drive you completely mad if you attempted anything remotely complex with it. It also cries out for namespaces.

The approach is for any Gui object (such as a form) to accept a shared variable for every property or state; for example to set the title and give focus to a form:

```
title←'My Window'        ⍝ Set the title
see←1                    ⍝ Make it visible
focus←1                  ⍝ Set keyboard focus on it
```

... assuming all the shares have been set up in advance. The process of building a Gui object becomes

```
handle ← CreateDlg template
⍝ then share a bunch of variables
⍝ then assign them
see←1      ⍝ finally show it
```

The templates are built by an external designer, copied to the OS/2 clipboard and pulled into the workspace as arbitrary character vectors suitable for the `CreateDlg` function.

Jim's closing message was "Windows programming is as simple as ←, so go and tell the boss". I wish he was right!

GENERAL ARTICLES

Readers might like to note that the J scripts printed at the end of David Benn's implementation of turtle graphics are available via FTP from:

/pub/ACE/otherlang/J/tg

on:

ftp.appcomp.utas.edu.au

As well as having fun with the 3D noughts and crosses game, the more enterprising reader might like to use the projection code to reproduce some of the figures included in the article. These required an initial projection of a wireframe cube, followed by a second projection of the projected image on to the screen / paper. The viewing angle was chosen by spinning this final image around until it looked right, then snapping the result to the clipboard via a Dyalog metafile, and then pasting it into CorelDRAW! to adjust line weights, and label the vertices.

The game will be included with Dyalog APL 7.2 if you have can't face typing in the code. Of course you can still enter the competition as you don't need the 3D visualisation just to think up a winning algorithm. Hopefully, Dyadic will be able to host a tournament at Lancaster.

Font For The Future

by F.H.D. van Batenburg
Email: sbqbeb@rulsfb.leidenuniv.nl

It was a pleasant surprise to read Bill Chang's paper [1] and to learn that WWW uses 8 bits for file transfer which would enable us to transmit APL characters. If only we could agree over a proper APL character set and mapping.

His proposal to agree on a common positioning of the APL characters seems very important to me. I wish we could all agree quickly on this so we can exchange code electronically and read it easily. This might raise the discussion how "easy" an ASCII transliteration scheme is. I think that an agreed ASCII transliteration is very useful and will remain so as long as people work with hardware or software that cannot display any particular chosen font. And that will still be the case quite a few years from now. However for me the APL characters are the easiest to read and any ASCII transliteration is a second best choice that is forced upon me by the hardware or software. So when my hard- and software allow it, I prefer the APL symbols.

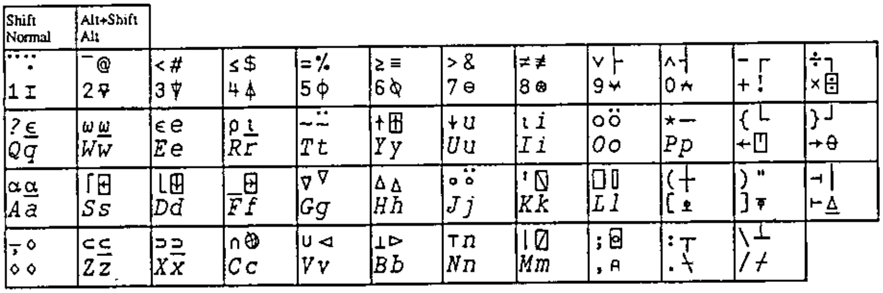
In his comments in *Vector* 12.2 [2], Adrian Smith stresses his need to cut and paste between his Web browser, word processor and APL program (Dyalog and Manugistics). Again a plea for a quick decision for a universally agreed mapping of APL characters.

However much I can agree over the importance of such a "universal agreed-upon mapping" of APL characters, I would like to bring up another requirement that might conflict with the above-mentioned cut/paste requirement. ***I want to enter APL in my word processor just as if I am using the APL 2741 keyboard.*** This means that by pressing key <3>, I would like to see "3" in my text, by pressing <Shift+3> I would like to see "<", pressing <Alt+3> should display "#", and pressing <Alt+shift+3> should display "⍒".

As far as I know, on the PC (Windows 3.1) the position of the characters in the fontfile determines their particular keys. There is no user-definable mapping table to associate a particular character in the fontfile to a specific Alt/Shift/key combination. So there is no way to tell Word that pressing key <A> or key-combination <Shift+A> should bring anything else forward but character 96 in fontfile APL, respectively character 66 (origin 1). Or is there?

The same holds true for the Macintosh; yes, I can switch to another generic mapping (for example "German-mapping" instead of "USA-mapping") but this will work for all subsequent fonts and is not specific to a particular APL font, or within a particular application. I know that Unix enables font-keyboard mappings, and that Dyalog APL does so, but I don't know of any way to do that in Word for Windows or in Word on my Macintosh.

So based on the assumption that there is a 1:1 relationship between the character position in the font file and the position on the keyboard, I defined a TTF font mapping on the Macintosh, using Fontmonger with the font that Vector provided (thank you Adrian) that makes it possible for me to type in all APL text as if it was a real 2741 typewriter, using <Alt> and <Alt+Shift> combinations for "overstrikes".



Shift Normal	Alt+Shift Alt										
¨ · 1 ⌶	¯ @ 2 ⍫	< # 3 ⍒	≤ $ 4 ⍋	= % 5 ⌽	≥ ≡ 6 ⍉	> & 7 ⊖	≠ ≢ 8 ⍟	∨ ├ 9 ⍱	∧ ┤ 0 ⍲	- ┌ + !	÷ ┐ × ⌹
? ⍷ Q q	⍵ ⍹ W w	∊ e E e	⍴ ⍸ R r	~ ⍨ T t	↑ ⍐ Y y	↓ u U u	⍳ i I i	○ ö O o	⋆ ─ P p	{ └ ← ⎕	} ┘ → θ
⍺ ⍶ A a	⌈ ⍇ S s	⌊ ⍗ D d	_ ⍈ F f	∇ ∇ G g	∆ ∆ H h	∘ ⍤ J j	' ⍂ K k	⎕ ⌷ L l	(┼ [⍎	) "] ⍕	⊣ │ ⊢ ⍙
⍪ ◇ ◇ ◇	⊂ ⊆ Z z	⊃ ⊇ X x	∩ ⍝ C c	∪ ⊲ V v	⊥ ⊳ B b	⊤ n N n	| ⍁ M m	; ⍠ , ⍝	: ┬ . ⍀	\ ┴ / ⌿	

Figure 1: Keyboard mapping of Macintosh TTF font APL_2741.

The figure shows that typing is mostly according to the "APL keyboard". Lowercase keys yield APL capital characters, uppercase yield APL non-overstrike functions (yes, I know that the word "overstrike" has lost its meaning since we do not overstrike any more, but I still use it to indicate the more complex characers), and <Alt+...> yields the normal APL overstrikes. All very regular, in accordance with the APL keyboard specification.

Only the <Alt+Shift+...> combinations are a bit irregular. As you can see there are several places that map to the same character. For example "i" is associated with <Alt+I> as well as with <Alt+Shift+I>, and the same <Alt+...> and <Alt+Shift+...> convergence holds for "e", "n" and "u". This is a particular quirk of the "USA-keyboard" font-mapping of the Mac-O.S., so I have to put up with it.

For this reason I could not place the underscored epsilon "⍷" at <Alt+Shift+E>, underscored iota "⍸" at <Alt+Shift+I> nor "⍗" on the same key alongside the "↓", much as I would like to.

Furthermore <Alt+Shift+6>, <...+8>, <...+9> and <...+10> do not yield "^", "*", "(" and ")". The reason was that I did not want to put one character on two positions in the font. For example "*" is already defined by key <Shift+P> and the corresponding character is placed in position 81. Associating <Alt+Shift+8> also with an asterisk would result in another asterisk placed in font position 162. Searching in text for an asterisk would either bring forward the one in position 81 or the other one in position 162, but not both. This is of course not acceptable in text processing, unless the two asterisks have a very different appearance. Similarly for searching on caret "^" (yes, that one looks a bit different from APL's and-function "∧", but not that different as you can see) and opening and closing parentheses.

The <Alt+Shift+...> also includes the framing characters and four arrowheads (at <Alt+Shift+G>, <...+H>, <...+V> and <...+B>) that connect to the lines made by the framing characters.

With this font, typing a manual or a paper (this one for example) is very easy. As mentioned before, on the Mac one can apply the 2741 key-mapping in any text processor: Word, ClarisWorks, BBEdit, SimpleText or whatever. I don't need to run a parallel APL session from which to cut and paste into my text, so I can also give a manuscript to an "APL-phobic" secretary and ask her to type it in for me.

Although I have not yet ported this font to the PC, such an approach will work as well on the PC. The normal and shifted keys will yield APL caps and APL non-overstrike functions in all text processors. Unfortunately Word for Windows does not allow one to use <Alt+...> and <Alt+Shift+...> to access a particular character. However for this I used to apply utility *COMPOSE* (a utility that DEC put into the public domain) and typed <Specialkey>+∇+|in order to get "⍒" and <Specialkey>+○+\ to get "⍉". Here the overstrike concept came to life again.

What is the mapping that makes this work? **See Figure 2 opposite.**

This shows the character positions in font APL_2741 on the Macintosh. It is different from Bill Chang's proposal, unfortunately, but also has much in common. Positions 1–33 are not used, and positions 128–169 are used sparingly. The first 128 positions are very much in line with most other proposals. Positions 170–256 are different, however. This is done to agree with the Macintosh key-mapping; as the PC has no <Alt+..> and <Alt+Shift+..> mapping in Word, I do not expect this mapping to disagree with any Word for Windows on the PC.

APL_2741 Light

	¨	)	<	≤	=	>	]	∨	∧	≠	÷	,	+	.	/
0	1	2	3	4	5	6	7	8	9	(	[	;	×	:	\
¯	α	⊥	∩	⌊	∊	_	∇	∆	⍳	∘	'	⎕	|	⊤	○
⋆	?	⍴	⌈	∼	↓	∪	⍵	⊃	↑	⊂	←	⊢	→	≥	-
⋄	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*
P	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	{	⊣	}	⍪	✿
✿	⍶	✿	✿	✿	✿	✿	✿	✿	✿	✿	✿	a	c	✿	✿
✿	✿	✿	✿	✿	✿	✿	✿	✿	✿	✿	✿	✿	✿	✿	✿
t	≢	⍋	⍒	⍉	⊛	⊖	*s*	*r*	*g*	⍫	e	*u*	⌹	"	⍥
⌽	¬	⍝	⍀	*y*	*m*	*d*	*w*	—	*p*	*b*	⍱	⍲	z	⍕	*o*
⍎	⌶	*l*	*v*	*f*	*x*	*j*	⍙	|	⍎	█	▒	░	£	⍷	*q*
!	┌	⌷	└	θ	┘	⌿	◁	|	⍣	⍨	@	#	$	%	≡
&	├	┤	⍹	⍸	⍁	[illegible]	⍐	✿	✿	⍇	⍗	⍈	⊜	∆	⍤
⍂	▯	┼	⍡	⍢	▷	*i*	*n*	⌼	⊤	*h*	*k*	⊆	∇	⊇	⍨

Are all APL characters included?

- Yes. All characters that are mentioned in the ISO extended standard are included.

- No. Many APL characters that are entered in the UCS2 standard [5] and [6]) are not included. All non-overstrike APL characters are there, but many overstrikes such as quad-equal, quad-diamond, down-tack-underbar, quote-underbar, circle-underbar, leftshoe-stile, downshoe-stile and many others are missing. All of them are without meaning in the current APL systems that I know of (but my knowledge is limited in this respect).

- Yes. All *APL characters* that Jim Brown proposed in [3] and Bill Chang in [1] and Adrian Smith in [4].
- No. Not all the *non-APL* characters that Jim Brown, Bill Chang and Adrian Smith proposed. They included national characters like the pound and the yen character and many, many accented characters.

My proposition with respect to national characters and accented characters is: ***do not include those characters in the APL character set.*** If inclusion would be easy, like the relatively small load of 11 framing characters, we could include them, but as it is a big problem we should exclude them. All accented characters are simply too much to expect to live together with a full APL character set in a file of 265 minus 64 reserved positions. This does not mean that those characters are unimportant, but only that they create a problem that can easily be circumvented by asking the user to use his ordinary text font if he/she wishes to use special accented or national characters. Those fonts are particularly suited for the national characters. These characters just don't belong to the APL character set.

As long as the UCS2 character set (a proposal where each character is represented by 16 bits) is not implemented everywhere it would be useful to agree on one character set mapping with all the current APL characters. And even after general acceptance of the UCS2 such a characterset configuration could be useful as the APL characters in UCS2 are spread out over several tables [5] and I suspect that one derived table that contains just the APL characters will often be used internally.

Is this an answer to Bill Chang's proposal? Unfortunately not. The alternative mapping and different reasoning presented here do not solve the problem of "what mapping" to choose, but rather enlarge the problem. My purpose was to put forward another criterion for font-mapping: typing in a text processor.

It seems that we now have several alternative suggestions that are not compatible, and even mutually exclusive. Let me finish with those questions and ask the APL users to think (and write) about an answer.

1. What characters do we want in a set of 256 (minus 64)? Live APL characters? Also the "dead" ones? National characters? Accented characters? frame (single and double) characters? (My suggestion is to take the live APL characters and framing characters only as shown in Figures 1 and 2.)
2. Do we choose a mapping such that we can cut/paste between Windows text processors (I assume this implies any Windows Web-browser) and Dyalog APL?

3. Do we choose a mapping such that we can cut/paste between Windows text processors and Manugistics APL?

4. Do we choose a mapping such that we can cut/paste between Macintosh text processors and 68000 APL?

5. Do we choose a mapping such that we can type APL in our Macintosh text processors? (This would be my preference.)

6. Do we choose a mapping such that we can type APL in our Windows text processors?

I have no answer to these questions that is satisfying for everybody. On the contrary, some alternatives seem mutually exclusive to me. We could ask ISO to choose a character mapping. This would hand over this Gordian knot to the ISO committee to unravel, but I do think that it is better if we could come to an agreement together. So I am anxious to know your opinion.

Dr F H D van Batenburg
Section Theoretical Biology, E.E.W. of Leiden University
Kaiserstraat 63
2311GP Leiden
The Netherlands

Tel: +31 71 5274972; Fax: +31 71 5274900;
Email: sbqbeb@rulsfb.leidenuniv.nl

References

[1] Chang, Bill; *My Computers are APL Machines.* Vector Vol 12 No.2 October 1995, pp. 7–8

[2] Smith, Adrian; *Some Comments.* Vector Vol 12 No.2 October 1995, p.9

[3] Brown, Jim; *IBM APL2 Unicode Character Mapping;* 1993

[4] Smith, Adrian; *One Font to Rule Them All.* Vector Vol 11 No.2 October 1994, pp. 105–106

[5] ISO/IEC 10646-1; Table 1 2 39 40 41 42 46; 1993

[6] Clayton, L.; internal discussion paper N93-14 regarding APL characters in UCS2; 1993

Multiprecision Arithmetic: Part III

by John Sullivan

Preliminary

In the function listings that follow I have assumed that the basic functions as defined in Parts I and II of this series are in a namespace called *mp*. If your APL does not support namespaces the functions will be defined in the workspace in the ordinary way, and therefore you should omit the *mp*. before all references to them: for instance in function *Spsp*, below, line 3 will be

```
[3]    n←Fexec n
```

Some simple applications

As I mentioned in Part II, it is much easier to handle multiprecision numbers if they are represented by character strings. So the first thing we will do with our basic functions is cover them by functions that process these character strings. (They will also process numeric input, as a result of the behaviour of *Fexec*). There are four functions for addition, subtraction, multiplication and division.

```
    ∇ z←a ADD b
     z←0 mp.Ffmt⊃mp.Fadd/mp.Fexec¨a b
    ∇

    ∇ z←{a}SUB b
      →(0≠⎕NC'a')/Δ1
      a←0 0
     Δ1:z←0 mp.Ffmt⊃mp.Fsub/mp.Fexec¨a b
    ∇

    ∇ z←a MUL b
      z←0 mp.Ffmt⊃mp.Fmul/mp.Fexec¨a b
    ∇
```

The division function introduces a small addition to the way numbers can be represented. Because the function deals with floating point rather than integers, sometimes we need to add precision to the dividend, in order to get extra precision in the quotient. This is done by using the letter P: in addition to saying 1.234E5 for 1.234×10^5, we can also say 1.234E5P2 for 1.234×10^5 with 2 extra "digits" of precision, one digit being the size of the current radix of the calculation. This P format is sorted out in *DIV*, rather than by *Fexec*, so it only applies to this function.

```
      ∇ z←a DIV b;p;q;r
[1]     b←mp.Fexec b
[2]     →(∧/~a∊⎕AV)/Δ3
[3]     p←0 ⋄ →('P'∧.≠a)/Δ2
[4]     p←(1+r←a⍳'P')↓a ⋄ a←r↑a
[5]     →(0≠⍴p←(p∊⎕D)/p)/Δ1
[6]     p←0 ⋄ →Δ2
[7]    Δ1:⎕SIGNAL(2<⍴p)/16
[8]     p←⍎p
[9]    Δ2:a←mp.Fexec a
[10]    →(p≤0)/Δ3
[11]    a[0]←a[0]-p ⋄ a←a,p⍴0
[18]   Δ3:z←0 mp.Ffmt a mp.Fdiv b
      ∇
```

Examples:

```
      '98765432109876543210' MUL '12345678901234567890'
1219326311370217952237463801111263526900

      '98765432109876543210' DIV '12345678901234567890'
8.0000000729000006633900060
```

and the following, which may be compared (sorry, IBM!) with [3] p.52 (see also Vector, Vol.11, No.4, p.124):

```
      B←'2223333333344444445555555'
      C←'12345.67890123456789'
      (B MUL C) DIV B
12345.67890123456789
```

When you write applications using the multiprecision suite try to avoid using these functions, because of the overhead of repeatedly translating into and out of character format. However, they are useful when using APL in desk-calculator mode for development and testing work.

The Strong Pseudoprime Test

This is a development from Fermat's theorem (also known as Fermat's little theorem) which says that if n is prime then $a^{n-1} \equiv 1 \bmod n$ for $0 < a < n$. This also holds true for some composite numbers, which are therefore called *pseudoprimes to base a*. If these composites are pseudoprimes for all *a* then they are called *absolute pseudoprimes* or *Carmichael numbers*. Fermat's theorem is improved by Euler's criterion, which states that an odd composite number n is called an Euler pseudoprime for base a if $a^{(n-1)/2} \equiv (a/n) \bmod n$, with $(a,n)=1$, where (a/n) is Jacobi's symbol. Riesel[1], p.98, carries this a little further and arrives at the concept of a *strong pseudoprime,* which is defined as follows:

An odd composite number n with $n-1 = 2^s d$, d odd, is called a *strong pseudoprime for base a* if either $a^d \equiv 1 \bmod n$, or $a^{2^r d} \equiv -1 \bmod n$, for $0 \leq r < s$.

This function is executed in the root namespace of the workspace. Its left argument is the base, which defaults to 2 if it is omitted, and the right argument is the number to be tested, which can be given as a character string or as a multiprecision number in vector format. If the number fails the test it must be composite, so we return with zero. Of course, if the number passes the test it could still be composite, but we return 1 to indicate that it is a pseudoprime.

```
      ∇ z←{a}Spsp n;d;e;i;n1;q;r;s;sm
[1]     ⍎(0=⎕Nc'a')/'a←2'
[2]     n←mp.Fexec n
```

We are going to split n–1 as described above. We strip the powers of 2 from n-1, count them into s, quotient is d

```
[3]     n1←d←n mp.Fsub 1 ⋄ s←0
[4]     ⍎((d[0]>0)∨~2|¯1↑d)/'d←⊃d mp.Idiv 2 ⋄ s←s+1 ⋄ →⎕LC'
```

If $a^d \equiv 1 \bmod n$ then exit

```
[5]     →(z←0 1 mp.Fequal e←a mp.Impow d n)/0
```

We now look at the variable r. When r is 0 we test the number we have just generated.

```
[6]     →(z←n1 mp.Fequal e)/0
[7]     r←1
```

Square d as r increases. We are testing for –1 modulo n, which is equivalent to n–1 modulo n, since `Imod` returns positive results only

```
[8]   ∆2:→(z←n1 mp.Fequal e←(e mp.Fmul e)mp.Imod n)/0
[9]     →(s>r←r+1)/∆2
```

You may want to add some progress-report code to this function or the `Impow` function, because for large numbers they take some considerable time to run.

For an application of this theorem let us look at a repunit. As mentioned in my preamble to Part I, the repunits are the numbers of the form $(10^n-1)/9$. Obviously these can only be prime if n is prime, so let us look at the case n=17.

```
        Spsp 17ρ'1'
```

After about fifteen seconds (on a 486/33) comes the result (0, implying composite), so either we give up on this one, or we start trying to factor it. (The factors are in fact 2071723 and 5363222357, see later on, and also Yates[2], p.142.)

A primality test

It is not my intention to regurgitate the contents of chapter 4 of Riesel here, but this function is included as an example of how to write APL code to fit the theorems in the book. This function is based on Theorem 4.6, p.109, and uses a relaxed converse of Fermat's theorem to determine the primality of an integer.

The prerequisites for this test are as follows. If N is the number to be tested we must have a partial factorization of $N - 1$, such that if $N - 1 = R\ F$ (where F is the factored part and R is the remaining part) then $R < F$. The theorem states that if we can find an integer a such that $\mathrm{GCD}(a^{(N-1)/q} - 1, N) = 1$ for each divisor q of F, and $a^{N-1} \equiv 1 \bmod N$, then N is prime. This last condition implies that N is a pseudoprime to base a: this is not tested in the function here, because it takes so long, and it is assumed that the first thing that anyone does in testing for primality is a pseudoprime test, so it will have been tested already.

This function takes a as its left argument (which defaults to 2 if not given). The right argument is a 2-element vector of N and F in any format you care to supply (character, scalar or multiprecision number). The result is a return code followed by a helpful message: the return codes are 1 for a prime, -1 for a composite number and 0 if the test is inconclusive, in which case you can repeat it with other values of a.

We start off by sorting out our input and preparing to loop through the divisors. Note that we only really need one copy of each divisor in F, not all of them. Also, we do not test that each factor of F actually divides $N - 1$ because we assume that these details have been sorted out before calling the function.

```
      ∇ Z←{A}Test4_6 X;A;B;F;J;N;N1;Q;R;⎕IO
[1]     ⎕IO←0
[2]     N F←X ⋄ ⍎(0=⎕NC'A')/'A←2'
[3]     N←0 0 mp.Fadd mp.Fexec N ⋄ N1←N mp.Fsub 0 1
[4]     F←0 mp.Fadd¨∪mp.Fexec¨F
[5]     J←0
```

Now test `1=GCD((B←¯1+A*N1÷F[J]),N)`. We can shorten the calculation here by noting that GCD(*B*,*N*) is the same as GCD(*B* mod *N*,*N*) and the mod bit is done in *Impow*, so if *B* mod *N* is 0 or 1 then the GCD is *N* or 1. The function is verbose here because it takes a fair length of time to run, and we want to know what is going on.

```
[6]   Δ1:Z←'Testing ',(0 mp.Ffmt J⊃F),', exponent is ',0
mp.Ffmt B←⊃N1 mp.Idiv J⊃F
[7]     Z,' residue is ',0 mp.Ffmt B←0 ¯1 mp.Fadd A mp.Impow B N
```

If *B* mod *N* is 0 the test was inconclusive, if it is 1 then this part of the test has succeeded. If it is anything else we can continue to calculate the GCD. If this is 1 then we continue with the next factor of *F*, otherwise we have found a factor of *N*, which must be composite.

```
[8]     →(0 0 mp.Fequal B)/Δ3    ⍝ Test inconclusive
[9]     →(0 1 mp.Fequal B)/Δ2    ⍝ GCD must be 1, so try next one
[10]    →(0 1 mp.Fequal Q←N mp.Gcd B)/Δ2  ⍝ Else get GCD.
[11]    Z←¯1('Composite! ',(0 mp.Ffmt Q),' is a factor.') ⋄ →0
[12]  Δ2:→((⍴F)>J←J+1)/Δ1
[13]    Z←'Prime. By Riesel Th. 4.6 with factor',(1<⍴F)/'s'
[14]    Z←dbr Z,,⎕FMT 0 mp.Ffmt¨F ⋄ Z←1 Z ⋄ →0
[15]  Δ3:Z←0('Riesel Th. 4.6 primality test inconclusive with
A=',⍕A)
      ∇
```

The GCD function uses Euclid's algorithm to calculate the Greatest Common Divisor. (It is stored in namespace mp.) In order to make things a bit easy when scalars are supplied, there are two paths through the function.

```
      ∇ Z←A Gcd B
[1]     →(1∧.=(⍴A),⍴B)/Δ2
[2]     ⍎(B Fgt A)/'A B←B A'
[3]   Δ1:→(0 1 Fequal Z←A Imod B)/0
[4]     ⍎(0 0 Fequal Z)/'Z←B ⋄ →0'
[5]     ⍎(0 Fgt Z)/'Z←Z Fadd B'
[6]     A←B ⋄ B←Z ⋄ →Δ1
```

The following code replicates the previous code, but for scalar rather than multiprecision input.

```
[7]    Δ2:⍎(B>A)/'A B←B A'
[8]    Δ3:→(1=Z←B|A)/0
[9]      ⍎(0=Z)/'Z←B ⋄ →0'
[10]     ⍎(0>Z)/'Z←Z+B'
[11]    A←B ⋄ B←Z ⋄ →Δ3
      ∇
```

For an example we can test $N = 39 \times 2^{22} + 1 = 163577857$, which can easily be verified a prime by trial division.

```
      2 Test4_6 163577857 2
Testing 2, exponent is 81788928, residue is 0
0  Riesel Th. 4.6 primality test inconclusive with A=2

      3 Test4_6 163577857 2
Testing 2, exponent is 81788928, residue is 0
0  Riesel Th. 4.6 primality test inconclusive with A=3

      5 Test4_6 163577857 2
Testing 2, exponent is 81788928, residue is 163577855
1  Prime. By Riesel Th. 4.6 with factor 2
```

To show what can happen, I will give an example with a small Carmichael Number (which, as I said earlier, is pseudoprime for all bases). Note that the strong pseudoprime test is stronger than that used for the Carmichael numbers: as 294409 (= 37 × 73 × 109) is not a strong pseudoprime to base 2 we would not normally have got as far as testing it with this function.

```
      Test4_6 294409 (2 3 29 47)
Testing 2, exponent is 147204, residue is 0
0  Riesel Th. 4.6 primality test inconclusive with A=2

      3 Test4_6 294409 (2 3 29 47)
Testing 2, exponent is 147204, residue is 0
0  Riesel Th. 4.6 primality test inconclusive with A=3

      5 Test4_6 294409 (2 3 29 47)
Testing 2, exponent is 147204, residue is 32264
¯1  Composite! 4033 is a factor.
```

Factorizing

There are several different methods for finding the factors of a number. The obvious first choice for a method is trial division: keep dividing your number by successive primes until you get a zero remainder: divide the number by that prime and start the process again on the quotient with the same prime (it may appear more than once in the factorization) until the next prime is greater than the square root of the remaining number. This works fine for small numbers (with a computer this process is relatively quick) but is no good for numbers like $(10^{33}-1)/9$, where the divisors 67^2, 271, 397 and 126951 are found very quickly, but the remaining number is a bit more difficult to factorize (it is 213524917 × 68618940391). This trial division can be speeded up in certain cases, for example the primitive prime factors of $(a^n - b^n)/(a - b)$ must be of the form $kn+1$ (or $2kn+1$ if n is odd), which eliminates a lot of checking, but there's still a lot left to do.

The next systematic factorizing method to appear, is due to Fermat. The idea is to write an odd composite number as the difference between two square numbers. If we can write $n = x^2 - y^2 = (x - y)(x + y)$ where $x - y \neq 1$, we immediately have a factorization. See Riesel[1], p.153ff, for the details.

We start by making our input numeric, then taking its square root. If the remainder is 0 we have hit on a factorization immediately, so we exit; otherwise we compute an intermediate value $m = [\sqrt{n}] + 1$, which is the smallest possible value of x.

```
      ∇ r←Fermat n;a;m;sqrem;z
         n←mp.Fexec n
         a sqrem←mp.Isqrt n
         →(~0 0 mp.Fequal sqrem)/Δ1
         'Number is the square of ',0 mp.Ffmt r←a ⋄ →0
        Δ1:m←a mp.Fadd 0 1
```

We consider $z = x^2 - n$ and check whether this is a square. If it is then we have finished, otherwise we try the next possible x, i.e. $m + 1$, and try again.

```
         z←(m mp.Fmul m) mp.Fsub n
         a sqrem←mp.Isqrt z
         →(0 0 mp.Fequal sqrem)/Δ3
        Δ2:z←z mp.Fadd 1 mp.Fadd 2 mp.Fmul m
         m←1 mp.Fadd m
         a sqrem←mp.Isqrt z
         →(~0 0 mp.Fequal sqrem)/Δ2
        Δ3:r←(m mp.Fsub a)(m mp.Fadd a)
      ∇
```

This function is quick if the two factors are approximately the same size, and it is horribly slow if they are wide apart. It can be made to finish quicker by multiplying the number to be factored by small numbers, but how you decide which small numbers to use is a black art. I think most examples in the reference books are worked backwards from known results (I must confess that my examples were all worked backwards also!).

It can also be made to run much faster for numbers of a certain type. For example, if we know that all the factors of a number have the form $2kn + 1$ (which they do for most numbers of the form $a^n \pm b^n$ for example) then we can reduce the number of iterations by a factor of $2n^2$. See Riesel, p.189, for the details. The function is very similar to the previous one, but supply the value of n as the left argument (here called p to confuse everybody):

```
     ∇ r←p Fermatx n;a;m;mod;res;sqrem;t;x;z
        n←mp.Fexec n
        a sqrem←mp.Isqrt n
        →(~0 0 mp.Fequal sqrem)/Δ1
        'Number is the square of ',0 mp.Ffmt r←a ◇ →0
```

A bit of number theory comes in here. We calculate the modulus ($2p^2$) and residue of $(n+1)/2$ modulo $2p^2$. The residue must be congruent to 1 modulo p, and we have an error if it isn't.

```
       Δ1:res←(⊃(1 mp.Fadd n)mp.Idiv 2)mp.Imod mod←2×p×p
        ⎕SIGNAL(1≠p|1↓res)/11
        t←⊃(1 mp.Fadd a)mp.Idiv mod
```

The calculations are slightly different here, as we calculate an intermediate variable x to assist in the calculation of z. In order to avoid repetition of coding, an if-then-else block is provided to generate z differently first time round compared with further execution (the first time is identified by $z = 0$). m is only calculated if it is required to generate the next version of z. When we have finished we use x to obtain the factors.

```
        z←0 0
       Δ2:x←res mp.Fadd mod mp.Fmul t←1 mp.Fadd t
        →(0 0 mp.Fequal z)/Δ3
        z←z mp.Fadd m ◇ →Δ4
       Δ3:z←(x mp.Fmul x)mp.Fsub n
       Δ4:a sqrem←mp.Isqrt z
        →(0 0 mp.Fequal sqrem)/Δ5
        m←(mod×2)mp.Fmul x mp.Fadd mod÷2
        →Δ2
       Δ5:r←(x mp.Fsub a)(x mp.Fadd a)
     ∇
```

I tried to factor $(10^{17}-1)/9$, a number that we know is composite, see earlier. I seemed to be getting nowhere, so I tried multiplying it by $34k+1$ for some values of k. All values seemed to give the same result, i.e. a long wait for nothing, until I tried 2585 ($k = 76$). Then I got the following, in exceedingly quick time:

```
      0 mp.Ffmt¨17 Fermatx 2585 mp.Fmul mp.Fexec 17ρ'1'
 5355403955  5363222357
```

From this it is easy to see which factor should be divided by 2585 to obtain the factors of 11111111111111111.

Of course, this is a trivial example: normally the numbers will be much bigger and take much longer to factor. There are many other methods of factorizing large numbers, and the best method may be a combination of several of them running concurrently, the task terminating when a factor is found in any of them.

I hope that this article has been useful in showing how to apply the multiprecision functions to solve problems with primality testing and factorizing.

References

[1] Hans Riesel, *Prime numbers and computer methods for factorization* (Birkhäuser, 1985).

[2] Samuel Yates, *Repunits and Repetends* (Star Publishing, Florida, 1982).

[3] *The APL Handbook of Techniques* (IBM DP Scientific Marketing, 1988. Order No. S320-5996-0).

Books on Number Theory

In this article, I have mentioned several items from number theory which there is no room to explain. I would refer the reader to any standard text on number theory for further information, for example:

G H Hardy and E M Wright, *An introduction to the theory of numbers*, Oxford University Press, 1938 and several later editions. I am sorry to say that the index in this book is atrocious, and is unworthy of a standard textbook.

W Sierpiński, *Elementary theory of numbers*, North-Holland, 1988.

D M Burton, *Elementary number theory*, Allyn and Bacon, 1980. This is the Open University set book, and its explanations are very clear.

Turtle Graphics and J

by David J. Benn
D.Benn@appcomp.utas.edu.au

Introduction

Soon after starting to learn J in earnest, I encountered a book called *Fractals Visualisation and J* [Reiter95]. Having long held a fascination for chaos theory and fractals, I was interested - and somewhat relieved - to note that J had been seriously used for the creation and exploration of fractal geometry, at a time when I was sceptical about the benefits of J over other languages.

Having studied J in the context of a coursework Masters unit on numerical computation, I have glimpsed something of the power of the language. It is an uncommon thing to find a powerful functional language which does not require exorbitant resources and at the same time provides good support for system-level programming in general and the Windows API and Macintosh Toolbox in particular. Indeed, I have moved from outright scepticism, to grudging respect, to admiration of J in only a few months of working with the language.

While looking through a number of issues of *Vector* recently, I came across Reiter's paper on fractals and J. [Reiter94]. Upon seeing a picture of the Koch curve (fractal snowflake) I was reminded of a Turtle Graphics program I had written to render this appealing shape. It occurred to me that it might be interesting – and even useful – to provide some means for doing Turtle Graphics in J.

Turtle Graphics

Turtle Graphics (TG) has its origins with the Logo language, created in the late 1960s by a team led by Seymour Papert [Papert80]. Logo was invented as a vehicle to help children think about problem solving [Wilson93] and for the interactive exploration of mathematics [Abelson92].

Logo can be considered to be a dialect of Lisp [Wilson93] and implementations are characterised by the following:

> Logo is highly interactive, organises programs as collections of procedures, includes lists as first-class data objects, and has built-in turtle graphics.

Imagine that you have control of a little creature called a turtle that exists in a mathematical plane or, better yet, on a computer display screen. The turtle can respond to a few simple commands: FORWARD moves the turtle, in the direction it is facing, some number of units. RIGHT rotates it in place, clockwise, some number of degrees. BACK and LEFT cause the opposite movements. [Abelson92]

While some early Logo systems used a dome-shaped robot (attached to a computer) which moved about on the floor, drawing on a large sheet of paper, most Logo systems represent the turtle as a triangle on a computer screen. In some systems, there is no turtle of any kind, merely the *notion* of one, for example [Benn95]. Where speed of rendering is important, in TG systems which do provide a visual turtle, the turtle may be hidden.

More recent versions such as Berkeley Logo take advantage of the windowing capabilities of modern operating systems (Windows, Macintosh) [Harvey93].

Turtle Procedure Notation

Abelson and DiSessa have the following to say in Appendix A to their book *Turtle Geometry*:

"Turtle Procedure Notation is meant to be a consistent and readable way to describe turtle programs independent of the quirks, details, and limitations of common computer languages, yet in such a way that the programs can be readily translated into whatever computer language is available." [Abelson92]

The authors used Logo for all the TG work done in developing the material for their book and in fact suggest that Logo is, except for minor variations in syntax, an actual implementation of Turtle Procedure Notation.

They discuss the basic operations and language features necessary for TG. These include features common to all programming languages, such as expression evaluation, integer and real data types, variables, procedures/functions, conditional evaluation (if...then...else), iteration and recursion.

Other language features such as a list datatype and delayed evaluation while arguably useful are not essential. Having said that, delayed evaluation *was* found to be useful when creating the **repeat** command for the current J implementation, while lists (vectors) were of great help in the maintenance of information about TG activity. Both are detailed below in the section entitled *Turtle Graphics Extension for J*.

They also discuss the relative merits of a number of commonly available programming languages as platforms for TG.

Circa 1980 BASIC is dismissed by the authors of *Turtle Geometry* as being unsuitable for TG due to a lack of support for modularisation, local scope, and recursion. This indicates that while the book has been reprinted several times, it has not been updated to consider modern BASIC dialects which *do* support such features. In fact, circa 1990 BASIC has more in common with Pascal and other modern imperative languages than with these early offerings.

One such example [Benn95] includes inbuilt TG commands, C-like structures, recursion, and sub-programs which double as procedures and functions. It is in any case encouraging to see that the authors of J for Windows recognise the need for inter-operability with languages such as Visual BASIC. [Pountain95]

Lisp, Smalltalk and APL all receive high marks from Abelson as languages which are well suited to the implementation of TG as language extensions. In particular:

> APL is quite suitable for turtle geometry. It is procedure-oriented, with inputs and outputs. It allows "environment building" in the way Logo does. Furthermore, it is recursive and handles compound data, such as vectors, in the cleanest fashion of any language mentioned here.
>
> We...know of at least one major installation that has turtle commands built into APL. [Abelson92]

The parallels to J in the above commentary are obvious.

Turtle Graphics Extension for J

Procedure definitions can be thought of as a way of extending the set of operations in a programming language [Abelson92], and since J provides the means by which functions can be defined and stored in scripts, such extension is supported by J. But at least as important is the degree to which a language permits interactive environment building in an experimental fashion, as opposed to the edit-compile-test cycle of languages such as Pascal and C (since programs written using them are most often compiled). Like Logo, J is quite suitable for this style of programming.

In setting out to implement Turtle Procedure Notation in J, it was determined that greater use would have to be made of J's control structures and explicit function definition than is perhaps customary. Purists who believe these to be largely unnecessary may consider the extent to which they have been used here excessive, but in reality the primitive commands and operations required for TG are essentially procedural in nature.

It should come as no surprise that J's Window Driver was also used extensively. A close examination of the code will however reveal smatterings of more traditional J code. The code was implemented and tested under J FreeWare for Windows. It was later also tested under the Standard edition (v2) of J for the Macintosh.

The current J implementation, hereafter referred to as JTG, supports multiple windows with one imaginary turtle per Window. Up to MAXWDW (currently defined as 10) windows may be open at once, allowing one to have multiple images in various stages of completion. TG operations are therefore directed at particular windows. This is facilitated by making the first parameter of each command or operation a number from 1 to MAXWDW indicating the window-id.

The following information is maintained about each window via J lists: whether or not the Nth window is open, whether the pen is up or down in each window (presumably stemming from the original turtle robots which lifted a pen up and down so that drawing would occur only when desired, but nevertheless useful), the heading in degrees for the turtle in each window, and the current x,y coordinates for each window's turtle. J verbs such as *head, behead, tail, amend* and *from* are used extensively for the maintenance of these lists.

JTG consists of a number of functions intended for internal use by a set of "public" interface functions. The internal functions will not be discussed here but can be found in Appendix 3, with comments. The interface commands/functions with the necessary parameters are as follows:

openwdw wdw
closewdw wdw
wait wdw
clear wdw
home wdw
setxy wdw x y
xycor wdw
penup wdw
pendown wdw
setheading wdw degrees
heading wdw
left wdw degrees
right wdw degrees
forward wdw distance
back wdw distance
n **repeat** commands

This list constitutes the essential commands and functions found in the specification for the Turtle Procedure Notation of [Abelson92] in addition to commands for handling windows. For a more detailed description of each of the above, see Appendix 1.

JTG Example Interaction

Here are some examples of the kind of interaction possible with JTG. What follows assumes that you have executed the functions in Appendix 3 on a suitable J system.

Consider what happens when typing the following code one line at a time. A space will be left between each line by JTG. After the **openwdw** command has been issued you will need to reselect the *jx* window into which the commands are being typed:

```
openwdw 1
forward 1 300
right 1 90
forward 1 300
right 1 90
forward 1 300
right 1 90
forward 1 300
```

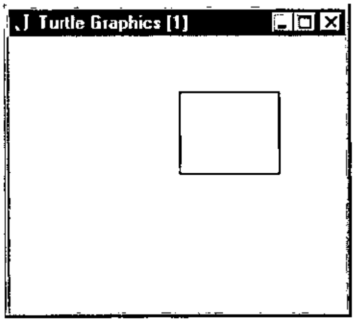


Figure 1 - humble beginnings with JTG.

This opens a window called "Turtle Graphics [1]", moves the turtle forward 300 steps, rotates the turtle right by 90 degrees, and then repeats the last two steps 3 more times. By this time there is a 300x300 square in JTG window 1. For such repetition, JTG provides the **repeat** function, which is essentially the same as that provided by Logo, differing only in syntax. Try the following sequence:

```
clear 1
4 repeat 'fd 1 300' then 'rt 1 90'
```

This clears the previously opened window, and draws a square in it, this time with a single line of code. The results are the same as in Figure 1. Note that all windows were resized before being copied and pasted so as to take up less space in the body of this paper.

Next, try the following code:

```
clear 1
penup 1
setxy 1 _50 _350
pendown 1
9 repeat 'fd 1 700' then 'rt 1 160'
closewdw 1
```

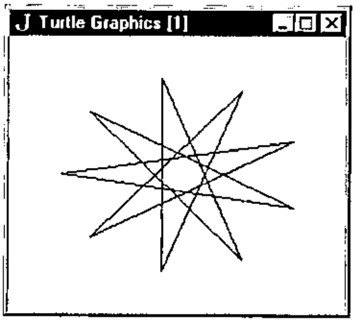


Figure 2 - A star is born.

The window is cleared again, the turtle is repositioned and a star is plotted. The last command closes the window. Figure 2 shows the star.

It is also instructive to examine the return values of the **xycor** and **heading** functions at various stages, although no examples of this are shown here.

Appendix 2 contains more JTG examples, via explicit function definitions. Example usages are:

```
polygon 1 7 200
polygon 2 50 20
```

which open two windows - one after the other, these do *not* run concurrently however - creating a pentagon and a circle (or a polygon with enough sides to *look* like one, but try increasing the length of each side) respectively. The wait command is used to await window closure via the window close-box, and finally the window is closed automatically by the **closewdw** command. Figures 3 and 4 show these polygons.

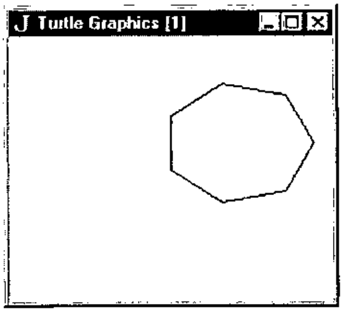


Figure 3 – a 7-sided polygon.

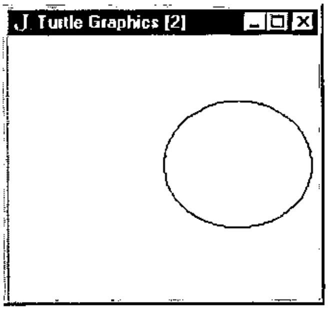


Figure 4 – a circle, almost.

Two TG favourites are the fractal tree and the fractal snowflake (Koch curve). Try the following:

```
tree 1 100          and
openwdw 1
penup 1
setxy 1 150 300
pendown 1
snowflake 1 3 600
closewdw 1
```

Notice that both the tree and snowflake procedures make good use of recursion. Figures 5 and 6 show the result of executing the above code. These fractals are described in [Abelson92] and a variety of other sources. Note the difference between the approach taken in the creation of the snowflake via TG as compared with Reiter's polygon approach [Reiter94].

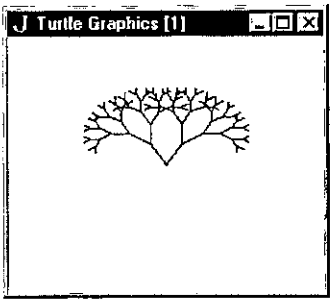


Figure 5 – a fractal tree.

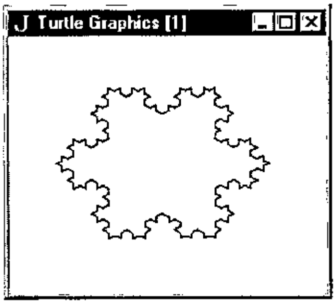


Figure 6 – a fractal snowflake.

For something a little different, one can easily create what appears to be a 3D torus by drawing a series of circles around a common centre. This is shown in Figure 7 and the code to do this is as follows:

```
36 repeat '24 repeat ''fd 1 60'' then ''rt 1 15''' then 'fd 1 3' then 'rt 1 10'
```

and assumes the existence of window 1.

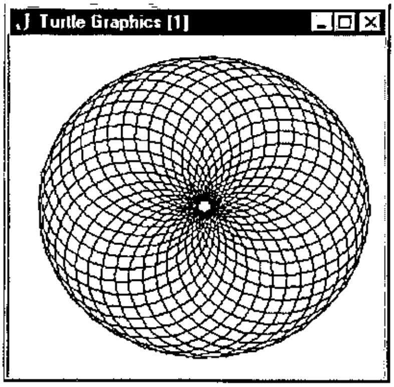


Figure 7 – what appears to be a 3D torus.

Last but not least, executing the following code: **flower 1 5** and assuming the existence of window 1 gives the result shown in Figure 8. The code for this can be found in Appendix 2. The flower function calls **petal** and **arcr** (arc right).

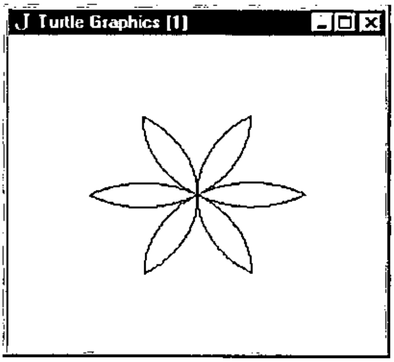


Figure 8 – beginnings of a flower.

Future Enhancements

A number of improvements are possible. For one thing, even on a 100 MHz 486DX4 running Windows 95, J Freeware for Windows executes JTG code quite slowly. Using the commercial Windows version might result in faster execution. JTG was also tested using the Macintosh Standard edition (v2) of J running on a Mac Centris 660AV, and approximately the same execution speed was noted.

As a rough comparison, the tree algorithm above takes several seconds to complete with JTG, while in ACE BASIC – a compiled language admittedly – on a 50 MHz 68030 Amiga 1200, the drawing takes less than a second.

Other reasons for the slow execution speed are:

- Each JTG interface function's first parameter is a window-id which is checked for validity each time the function is called. Enforcing the use of a **selectwdw** interface function (which already exists internally) and having JTG commands act upon the last window selected would remove much of this overhead. However, the beauty of having to specify a window for each JTG command/ function is that it is immediately obvious which window will be targeted.
- There is a certain amount of overhead involved in the maintenance of lists as outlined above, which would not exist if multiple windows weren't allowed.

A further overhead could be *introduced* if the number and nature of all parameters (not just the window-id) for each JTG command/function were to be checked before use. As it now stands, J will produce errors if the formal and actual parameters don't match.

One problem that should be attended to is that if a window is closed via its close-box, and no **closewdw** command is issued for the window subsequently, the window lists will contain incorrect information for that window.

Other possible enhancements include:

- A function which returns TRUE if a mouse button is depressed, or the close-box of a window has been activated. These would be useful for cutting a program's execution short.
- The repeat command could be enhanced to allow a test for a condition (e.g. mouse press) as well as just a count. [Abelson92]
- Being able to change a turtle's pen colour.
- Having JTG display error messages instead of just "beeping" when an error occurs (e.g. when a non-existent window is targeted).

Other possibilities (albeit probably less useful) include displaying the turtle, and multiple turtles per window. The latter would permit turtle interactions [Abelson92]. Such additions would be likely to slow execution even further.

One advantage of the current slow execution is that it is possible to watch images being drawn at a reasonable pace, so making it possible to understand the algorithm being carried out more easily than if the drawing were to be all over in a flash. In a faster compiled language, the rendering can of course be slowed by inserting delays between certain commands.

Deficiencies Noted in J

While the author is in general very impressed by J, a few problems were noted in creating JTG:

- Logical operators (eg. and (*.)) are not short-circuit operators. This means that instead of being able to say:

  ```
  if. inrange wdw and windowactive wdw do.
  .
  ```

  one is forced to write:

  ```
  if. inrange wdw do.
      if. windowactive wdw do.
  .
  ```

  or risk a run-time error, say, if the window-id is not in range. This may impact upon performance and increase the level of control-structure nesting required.

- Explicitly defined functions seem to be permitted only if parameters are supplied. This was a problem when attempting to make some functions/ commands parameterless for the purpose of removing the requirement of having each operation specify a window explicitly, as mentioned in the previous section, e.g.

  ```
          xycor
  instead of
          xycor wdw
  ```

  This may very well amount to no more than a deficiency in the author's understanding of J however.

- No support for concurrency. It would be nice to have multiple TG programs running at once, to test out more than one idea at a time. Note that this is different from having multiple windows open, only one of which can be in use by the programmer at a given time.

- No programmer-controllable user-break mechanism. Something like an event trapping or exception handling mechanism would be useful when there is reason to think that a program might migrate to fairy land.

Conclusion

Despite a few problems, J was found to be a most suitable platform on which to base TG. This exercise has served to reinforce the author's belief that J is a language to be reckoned with, a serious functional language which deserves recognition for its innovations, and serious consideration by software engineers who find themselves involved with numerical computation but also require inter-operability with the OS and other languages.

TG is a fun, powerful vehicle for understanding mathematics (in particular, geometry and certain fractal algorithms) which encourages an interactive, experimental approach, and like Logo, J provides a means - via appropriate extensions - to do this.

References

[Abelson92]	H. Abelson and A. DiSessa, 1992, *Turtle Geometry: The Computer as a Medium for Exploring Mathematics,* MIT Press: USA.
[Benn95]	D.J. Benn, 1995, *ACE Programmer's Guide and Language Reference,* http://www.appcomp.utas.edu.au/users/dbenn/
[Harvey93]	B. Harvey et al, *1993, Berkeley Logo,* Regents of the University of California.
[Iverson95]	K. Iverson, 1995, *J Introduction & Dictionary,* Iverson Software Inc.
[Papert80]	S. Papert, 1980, *Mindstorms: Children, Computers and Powerful Ideas,* Basic Books, Inc.
[Pountain95]	D. Pountain, September 1995, *The Joy of J,* BYTE Magazine, p 267.
[Reiter94]	C.A. Reiter, October 1994, *Fractals RYIJ,* Vector, vol. 11 no. 2, p 86.
[Reiter95]	C.A. Reiter, 1995, *Fractals Visualisation and J,* Iverson Software Inc.
[Wilson93]	L.B. Wilson and R.G. Clark, 1993, *Comparative Programming Languages,* Addison-Wesley.

Appendix 1: JTG interface command/function syntax and description

Each command or function is presented with the required syntax, and is followed by a brief description. This information was extracted from the commented code in Appendix 2. The command or function name is in **bold** print.

openwdw wdw

Open TG window (1000x1000 pixels) with specified numeric ID in the range 1..MAXWDW. The TG grid x,y range is -500..+500, -500..+500. The window contents are auto-scaling when window size is changed.

closewdw wdw

Closes specified TG window, amending lists.

wait wdw

Waits for some kind of message from the current window, typically a close-box click.

clear wdw

Clears specified window, resetting turtle position and heading to the defaults.

home wdw

Returns turtle to the home position, resetting turtle position and heading to the defaults. If the pen is down a line is drawn from the current to the home position.

setxy wdw x y

Sets the x,y position, drawing a line if the pen is down.

xycor wdw

Returns current x,y coordinate of turtle in specified window as a list.

penup wdw

Retracts the pen of the specified window's turtle.

pendown wdw

Lowers the pen of the specified window's turtle.

setheading wdw degrees

Changes the heading of the specified window's turtle.

heading wdw

Returns current heading of turtle in specified window.

left wdw degrees

Rotates turtle left by number of degrees specified. Note: a negative value will result in a right turn! (**left** may be abbreviated as **lt**).

right wdw degrees

Rotates turtle right by number of degrees specified. Note: a negative value will result in a left turn! (**right** may be abbreviated as **rt**).

forward wdw distance

Moves turtle forward by specified number of steps. Note: a negative distance will result in backward motion! (**forward** may be abbreviated as **fd**).

back wdw distance

Moves turtle back by specified number of steps. Note: a negative distance will result in forward motion! (**back** may be abbreviated as **bk**).

n **repeat** commands

Repeats the specified commands N times. The commands are expected to be laminated character strings. For example: *4 repeat 'fd 1 100' then 'right 1 90'*.

Scripts available via FTP!

I have just placed the files tg.js and tgeg.js (along with 2 readme files) in:

/pub/ACE/otherlang/J/tg

on:

ftp.appcomp.utas.edu.au

[message received on CompuServe 5th January 1996 – thanks, Ed]

Appendix 2: Example JTG Programs (tgeg.js)

Note: The script *tg.js* (Appendix 3) must be executed before this code is used.

```
NB.
NB. Examples of using TG from J.
NB. David Benn, Semester II, 1995.
NB.
NB. Execute the script tg.js before
running
NB. the following code.
NB.


NB. SYNTAX: polygon wdw sides length
NB.
NB. eg. polygon 1 4 200     [square]
NB.          polygon 1 50 20
        [circle]
NB.
polygon =. 3 : 0
wdw =. head y.
n =. head behead y.
length =. tail y.
degs =. 360 % n
i =. 0
openwdw wdw
while. i < n do.
  forward wdw , length
  right wdw , degs
  i =. i + 1
end.
wait wdw
closewdw wdw
)


NB. SYNTAX: tree wdw length
NB.
NB. eg. tree 1 100
NB.
drawtree =. 3 : 0
wdw =. head y.
length =. tail y.
if. length >: 20 do.
  right wdw , 30
  forward wdw , length
  drawtree wdw , length*0.75
  back wdw , length
  left wdw , 60
  forward wdw , length
  drawtree wdw , length*0.75
  back wdw , length
  right wdw , 30
end.
)


tree =. 3 : 0
wdw =. head y.
openwdw wdw
drawtree y.
wait wdw
closewdw wdw
)


NB. SYNTAX: snowflake wdw depth side
NB.
NB. eg. snowflake 1 3 600
NB.
NB. This assumes that window 1 is open
and the
NB. turtle has been suitably positioned.
NB.
koch =. 3 : 0
wdw =. head y.
depth =. head behead y.
side =. tail y.
if. depth = 0 do.
  forward wdw , side
else.
  koch wdw , depth-1 , side % 3
  left wdw , 60
  koch wdw , depth-1 , side % 3
  right wdw , 120
  koch wdw , depth-1 , side % 3
  left wdw , 60
  koch wdw , depth-1 , side % 3
end.
)


snowflake =. 3 : 0
wdw =. head y.
depth =. head behead y.
side =. tail y.
koch wdw , depth , side
right wdw , 120
koch wdw , depth , side
right wdw , 120
koch wdw , depth , side
right wdw , 120
)


NB. SYNTAX: flower wdw size
NB.
NB. eg. flower wdw 200
NB.
arcr =. 3 : 0
wdw =. head y.
r =. head behead y.
degs =. tail y.
```

```
i =. 1
while. i <: degs do.
  fd wdw , r
  rt wdw , 1
  i =. >: i
end.
)


petal =. 3 : 0
wdw =. head y.
size =. tail y.
  arcr wdw , size , 60
  right wdw , 120
  arcr wdw , size , 60
  right wdw , 120
)


flower =. 3 : 0
wdw =. head y.
size =. tail y.
i =. 1
while. i <: 6 do.
  petal wdw , size
  rt wdw , 60
  i =. >: i
end.
novalue
)
```

Appendix 3: Complete Source Code for JTG (tg.js)

```
NB.
NB. Turtle Graphics and J (JTG).
NB. Author: David Benn, Semester II,
1995.
NB.


NB.
NB. Global values.
NB.


TRUE =. 1
FALSE =. 0
novalue =. ' '
RADCONV =. 57.295779       NB. to convert
degs to radians.


hx =. 500
hy =. 500             NB. Turtle's x,y home
position at center of 1000x1000 grid.


DEGS =. 90            NB. Turtle's home
heading.


MAXWDW =. 10


UP =. 0
DOWN =. 1
FREE =. 0
TAKEN =. 1
FORWARD =. 0
REVERSE =. 1


penlist =. MAXWDW # DOWN
wdwlist =. MAXWDW # FREE
degslist =. MAXWDW # DEGS
txlist =. MAXWDW # 0
tylist =. MAXWDW # 0


NB.
NB. Miscellaneous functions.
NB.


wd =. 11!:0          NB. Window driver
foreign conjunction.
head =. {.
behead =. }.
tail =. {:
from =. {
amend =. }
and =. *.
not =. -.
sin =. 1&o.
cos =. 2&o.
execute =. ".
count =. #
then =. ,:           NB. See 'repeat'
definition.


NB. SYNTAX: beep n
NB.
NB. Sound the bell N times.
NB.
beep =. 3 : 0
wd 'beep ',(":y.),';'
)


NB.
NB. Internal turtle graphics functions.
NB.


NB. SYNTAX: inrange wdw
NB.
NB. Is specified window-id in the range
1..MAXWDW?
```

```
NB.
inrange =. 3 : 0
(y. >: 1) and (y. <: MAXWDW)
)


NB. SYNTAX: windowactive wdw
NB.
NB. Is specified window active (open)?
NB.
windowactive =. 3 : 0
((<: y.) from wdwlist) = TAKEN
)


NB. SYNTAX: penactive wdw
NB.
NB. Is pen of turtle in current window
down?
NB.
penactive =. 3 : 0
((<: y.) from penlist) = DOWN
)


NB. SYNTAX: move x y
NB.
NB. Move to x,y in currently selected
window.
NB.
move =. 3 : 0
wd 'gmove ',(":hx + head y.),' ',(":hy +
tail y.),';'
wd 'gshow;'
)


NB. SYNTAX: line x y
NB.
NB. Draw a line from current location to
x,y
NB. in currently selected window.
NB.
line =. 3 : 0
wd 'gline ',(":hx + head y.),' ',(":hy +
tail y.),';'
wd 'gshow;'
)


NB. SYNTAX: selectwdw wdw
NB.
NB. Make specified window the target for
TG commands.
NB.
selectwdw =. 3 : 0
wd 'psel TGparent',(":y.),';'
wd 'csel TG',(":y.),';'
)


NB. SYNTAX: changedegslist wdw degrees
NB.
NB. Change degslist entry for specified
window.
NB.
changedegslist =. 3 : 0
degslist =: (360 | (tail y.)) (<: head
y.) amend degslist
)


NB. SYNTAX: changexylists wdw x y
NB.
NB. Change txlist and tylist entries for
specified window.
NB.
changexylists =. 3 : 0
wdw =. head y.
x =. head behead y.
y =. tail behead y.
txlist =: x (<: wdw) amend txlist
tylist =: y (<: wdw) amend tylist
)


NB.
NB. Turtle graphics public interface
functions.
NB.


NB. SYNTAX: openwdw wdw
NB.
NB. Open TG window (1000 x 1000 pixels)
with specified
NB. numeric ID in the range 1..MAXWDW.
The TG grid x,y range
NB. is -500..+500, -500..+500. The window
contents are auto-scaling
NB. when window size is changed.
openwdw =. 3 : 0
if. inrange y. do.
  if. not windowactive y. do.
    wd 'pc TGparent',(":y.),';'
    wd 'pn "Turtle Graphics
[',(":y.),']";'
    wd 'xywh 0 0 180 150;'
    wd 'cc TG',(":y.),' isigraph;'
    wd 'csel TG',(":y.),';'
    wd 'gmove ',(":hx),' ',(":hy),';'
    wd 'pas 0 0;'
    wd 'pcloseok; pcenter; pscale; ptop;
pshow;'
    wdwlist =: TAKEN (<: y.) amend
wdwlist
        novalue
  else.
    beep 1
  end.
else.
  beep 1
end.
```

```
)


NB. SYNTAX: closewdw wdw
NB.
NB. Closes specified TG window, amending
lists.
NB.
closewdw =. 3 : 0
if. inrange y. do.
  if. windowactive wdw do.
    wd 'psel TGparent',(":y.),';'
    wd 'pclose;'
        Nth =. <: y.
    wdwlist =: FREE Nth amend wdwlist
        penlist =: DOWN Nth amend penlist
        degslist =: DEGS Nth amend
degslist
        txlist =: 0 Nth amend txlist
        tylist =: 0 Nth amend tylist
    novalue
  else.
    beep 1
  end.
else.
  beep 1
end.
)


NB. SYNTAX: clear wdw
NB.
NB. Clears window, resetting turtle
position
NB. and heading to the defaults.
NB.
clear =. 3 : 0
if. inrange y. do.
  if. windowactive y. do.
    selectwdw y.
    wd 'gclear;'
        move 0 0
        changedegslist y. , DEGS
        changexylists y. , 0 , 0
    novalue
  else.
    beep 1
  end.
else.
  beep 1
end.
)


NB. SYNTAX: home wdw
NB.
NB. Returns turtle to the home position,
resetting turtle
NB. position and heading to the defaults.
If the pen is down
NB. a line is drawn from the current to
the home position.
NB.
home =. 3 : 0
if. inrange y. do.
  if. windowactive y. do.
        selectwdw y.
        if. penactive y. do.
          line 0 0
    else.
      move 0 0
    end.
        changedegslist y. , DEGS
        changexylists y. , 0 , 0
    novalue
  else.
    beep 1
  end.
else.
  beep 1
end.
)


NB. SYNTAX: setxy wdw x y
NB.
NB. Sets the x,y position, drawing a line
if the pen is down.
NB.
setxy =. 3 : 0
if. inrange head y. do.
  if. windowactive head y. do.
        selectwdw head y.
    if. penactive head y. do.
      line behead y.
    else.
      move behead y.
    end.
        changexylists y.
    novalue
  else.
    beep 1
  end.
else.
  beep 1
end.
)


NB. SYNTAX: xycor wdw
NB.
NB. Returns current x,y coordinate of
turtle in specified window.
NB.
xycor =. 3 : 0
if. inrange y. do.
  if. windowactive y. do.
        Nth =. <: y.
        x =. Nth from txlist
        y =. Nth from tylist
        x , y
  else.
```

```
      beep 1
  end.
else.
  beep 1
end.
)


NB. SYNTAX: penup wdw
NB.
NB. Retracts the pen of the specified
window's turtle.
NB.
penup =. 3 : 0
if. inrange y. do.
  if. windowactive y. do.
    penlist =: UP (<: y.) amend penlist
    novalue
  else.
    beep 1
  end.
else.
  beep 1
end.
)


NB. SYNTAX: pendown wdw
NB.
NB. Lowers the pen of the specified
window's turtle.
NB.
pendown =. 3 : 0
if. inrange y. do.
  if. windowactive y. do.
    penlist =: DOWN (<: y.) amend penlist
    novalue
  else.
    beep 1
  end.
else.
  beep 1
end.
)


NB. SYNTAX: setheading wdw degrees
NB.
NB. Changes the heading of the specified
window's turtle.
NB.
setheading =. 3 : 0
if. inrange head y. do.
  if. windowactive head y. do.
        changedegslist y.
        novalue
  else.
        beep 1
  end.
else.
  beep 1
end.
)


NB. SYNTAX: heading wdw
NB.
NB. Returns current heading of turtle in
specified window.
NB.
heading =. 3 : 0
if. inrange y. do.
  if. windowactive y. do.
        (<: y.) from degslist
  else.
        beep 1
  end.
else.
  beep 1
end.
)


NB. SYNTAX: turn wdw degrees
NB.
NB. Rotate turtle by number of degrees
specified.
NB.
turn =. 3 : 0
if. inrange head y. do.
  if. windowactive head y. do.
        currdegs =. (<: head y.) from
degslist
        changedegslist (head y.) ,
(currdegs + tail y.)
        novalue
  else.
        beep 1
  end.
else.
  beep 1
end.
)


NB. SYNTAX: left wdw degrees
NB.
NB. Rotate turtle left by number of
degrees specified.
NB. Note: a negative value will result in
a right turn!
NB.
lt =. left =. turn


NB. SYNTAX: right wdw degrees
NB.
NB. Rotate turtle right by number of
degrees specified.
NB. Note: a negative value will result in
a left turn!
NB.
rt =. right =. 3 : 0
```

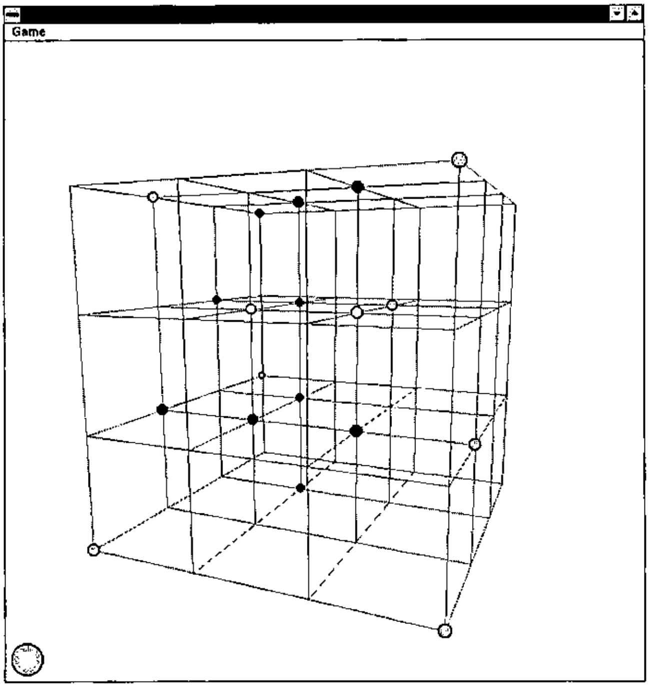
```
turn (head y.) , (- tail y.)
)


NB. SYNTAX: moveturtle wdw radius
direction
NB.
NB. Move current window's turtle by
specified amount
NB. (radius) forwards or backwards. The
turtle starts
NB. at the center of a circle and moves
to the perimeter.
NB.
moveturtle =. 3 : 0
wdw =. head y.
if. inrange wdw do.
  if. windowactive wdw do.
    radius =. head behead y.
    direction =. tail y.
    degs =. (<: wdw) from degslist
    if. direction = FORWARD do.
      theta =. degs % RADCONV
    else.
      angle =. 360 | (degs+180)
      theta =. angle % RADCONV
    end.
        curr =. xycor wdw
        currx =. head curr
        curry =. tail curr
    x =. currx + (radius * cos theta)
    y =. curry + (radius * sin theta)
    selectwdw wdw
    if. penactive wdw do.
          move currx , curry
      line x , y
    else.
      move x , y
    end.
    changexylists wdw , x , y
        novalue
  else.
    beep 1
  end.
else.
  beep 1
end.
)


NB. SYNTAX: forward wdw distance
NB.
NB. Move turtle forward by specified
number of steps.
NB. Note: a negative distance will result
in backward
NB.       motion!
NB.
fd =. forward =. 3 : 0
moveturtle y. , FORWARD
)
```

```
NB. SYNTAX: back wdw distance
NB.
NB. Move turtle back by specified number
of steps.
NB. Note: a negative distance will result
in forward
NB.       motion!
NB.
bk =. back =. 3 : 0
moveturtle y. , REVERSE
)


NB. SYNTAX: wait wdw
NB. Wait for some kind of message from
the current
NB. window, typically a close-box click.
NB.
NB.
wait =. 3 : 0
if. inrange y. do.
  if. windowactive y. do.
        selectwdw y.
        wd 'wait;'
  else.
        beep 1
  end.
else.
  beep 1
end.
)


NB. SYNTAX: n repeat commands
NB.
NB. Repeats the commands N times.
NB.
NB. The commands are expected to be
laminated
NB. character strings. For example:
NB.
NB.          4 repeat 'fd 1 100' then
'right 1 90'
NB.
repeat =. 3 : 0
:
i =. 1
while. i <: x. do.
  cmdlist =. y.
  while. (count cmdlist) > 0 do.
    execute head cmdlist
    cmdlist =. behead cmdlist
  end.
  i =. >: i
end.
novalue
)
```

Three-Dimensional Noughts and Crosses

by Duncan Pearson (causeway@compuserve.com)

This year's ***Vector Christmas Gift*** (albeit one month late) is a working version of the game of three-dimensional noughts and crosses. The game allows two players to play against each other on one computer. The ***Vector Christmas Competition*** (also late, and described in detail later) is to devise an algorithm to enable the computer to play.

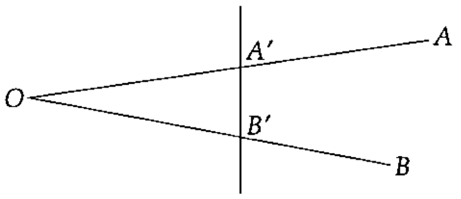


In the game a cubic wire frame with $n \times n \times n$ points is projected onto a window and using the keyboard the player can move the position from which it is seen up, down, left, right, in or out. The player selects a node on the cube with the mouse and a piece of his colour is placed on it. The turn then passes to the other player. The game is won when, by placing his counter, a player completes a straight line of n counters of his colour. A good game can be played with n equal to four. The game is not a trivial win for the first player but it is nonetheless possible for a better player to get a win, and some interesting tactics can be used.

I wrote the game primarily to understand how to project 3-d shapes onto a surface and most of the code, together with most of this article, is concerned with that. I will briefly explain the purpose of the game and then go into how I approach the problem of projection, finishing with the code and how that implements it. The code is written in Dyalog APL/W with ⎕*IO* zero and ⎕*ML* three (the highest APL2 compatibility available).

Projection

The problem is to make a two-dimensional picture appear to the observer as if it were a three-dimensional object. This means that the light rays coming from the picture into the eye of the observer[1] must follow the same paths as they would coming from the 3-d object.

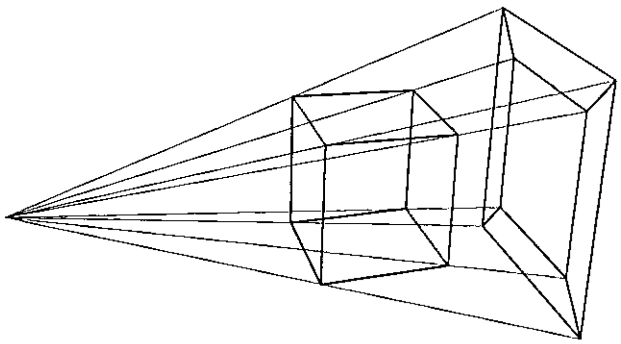


In the diagram, the light from A (part of the object) to O (the observer) is following the same path as the light from A′ to O, so as far as the observer is concerned A and A′ are the same point. Similarly BO and B′O follow the same path and so B and B′ are the same from the observer's point of view (literally).

If we had a 3-d shape which we wished to draw, a cube say with eight vertices, we could place a piece of card beyond the cube, make a line between the eye and each vertex, and extend them until they hit the card.

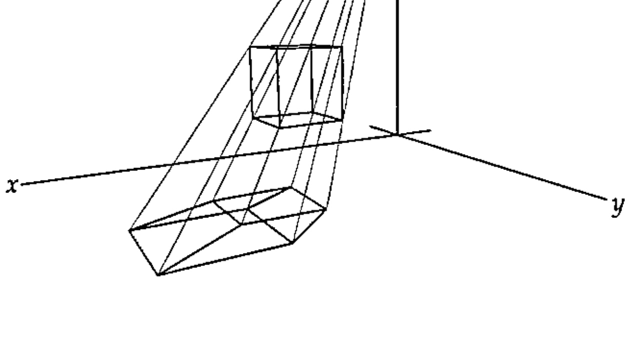
[1]Observers are assumed, for simplicity, to have had one eye removed or to be descendants of the legendary Cyclops. For a discussion of the effect of two eyes on the calculations see E.A. Clough, *Stereograms in J*, Vector 11.4 pp 110–116

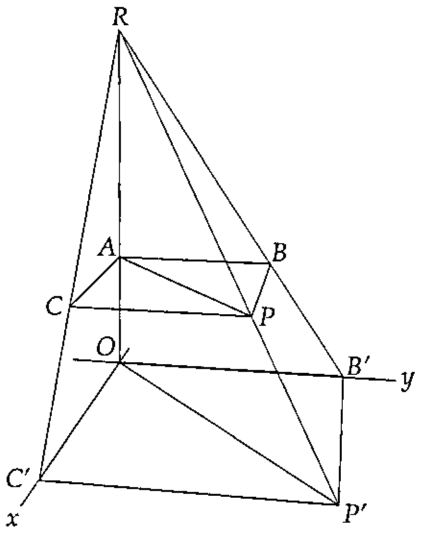
The eight marks on the card would be a projection of the cube onto the card and, if viewed from the position of the eye, would look like a cube:



To do the work on a computer we must calculate where the lines would pierce the card. The points of our 3-d object are represented by *(x, y, z)* co-ordinates as is the position of the observer. The card is a plane (which in co-ordinate geometry is represented by a linear equation in x, y and z) and we must find where the lines between the object points and the observer point intersect the plane.

The algebra involved is normally quite convoluted but there is one case where it is much simpler. If the observer is sitting on the Z axis (x=y=0) and the card is the XY plane (z=0) then we have the situation shown below:





Taking one point of the object P (at *(x, y, z)*) and looking at its projection P′ (at *(x′, y′, 0)*) in the XY plane, with the observer at the point R *(0, 0, r)* we can draw two similar triangles RAP and ROP′. As the triangles are similar the ratio of the distances RP′ to RP is the same as the ratio of RO to RA. That ratio is *r/(r-z)*.

Taking the similar triangles RBP and RB′P′ we have the ratio B′P′ to BP the same as RP′ to RP; but the length of BP is *x* so the length of B′P′, the X component of P′, is *xr/(r-z)*. Clearly the same holds true for C′P′, the Y component, and so we have the position of P′ as *(xr/(r-z), yr/(r-z), 0)*.

The procedure of projecting a wire frame object is to project the co-ordinates of the vertices onto the plane then to join on the plane the projections of any pairs of points that are joined in the object.

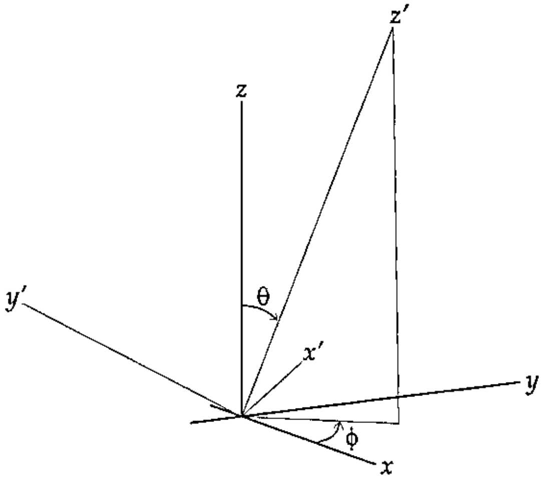
This is shown in the functions `Draw` and `DrawPoints` below.

```
 Draw
⍝ Project points from the current position then redraw the lines and circles
 projected←theta phi Transform points
⍝ Find the order in which to draw the points, those at the back first
 order←⍋projected[;2]
⍝ Calculate the size for the circles
 size←0.05×R÷R-projected[;2]
⍝ Project onto the XY plane
 projected←projected[;0 1]×2/,[0.1]R÷R-projected[;2]
⍝ Draw the wire frame
 'form.lines'⎕WS'points'(⊂[1 2]⌽projected[lines;])
⍝ Draw the taken points
 DrawPoints

------------------------------------------------------------------

 DrawPoints;lv
⍝ Draw the taken points
 lv←×colour[order]
 'form.points'⎕WS('points'((1.2-↑⌽⍴wins)ref ⌽lv⌿projected[order;]))
                 ('fillcol'((player,lv/colour[order])⌽¨⊂0 255 0))
                 ('radius'(0.15,lv/size[order]))
```

This function has the added complication of needing to adjust the radii of the circles representing the counters. For relatively small circles it is a good approximation that the size of the circle increases in the same ratio as the X and Y co-ordinates of its centre[2].

So, we can now draw any object as though we were looking down on it from above the origin. To draw it seen from another position we need to make a modification before we do the projection. The co-ordinates of the points of the object are given relative to a particular fixed set of axes. There is no reason why, for the purpose of the projection, we shouldn't choose our own axes and change the object co-ordinates so that they are relative to the new axes.

If we choose a new Z axis from the observer to the origin, and a new X axis perpendicular to it and in the old XY plane, then the direction of the new Y axis is defined as perpendicular to both of these:



In the game (remember the game) we move the observer not by specifying his co-ordinates relative to the base axes, but by giving (or rather by changing) his latitude and longitude and his distance from the origin. This gives, in the game, an

[2]This is only inaccurate with very large balls, where it is the solid angle subtended by the ball that is what must be represented, or with balls at the periphery of the observer's vision that need to be represented on the flat plane as ellipses. Neither of these conditions is true of the game.

impression of moving smoothly in a circle round the object, or of moving into or out from the object. So the observer is defined by two angles and a distance *r*, relative to the original axes. The first angle, which I have called *theta* (θ) throughout, is the angle between the fixed Z axis and OR, the line between the origin and the observer. The second angle *phi* (ϕ) is the anticlockwise angle between the fixed X axis and the projection of OR in the fixed XY plane.

These quantities (r, θ, ϕ) are known as the spherical polar co-ordinates of the observer.

Transforming the Co-ordinates

In order to get the X co-ordinate of a point relative to the new X axis we need to take the scalar product (using `+.×`) of its co-ordinates (relative to the old set of axes) with the co-ordinates of a point one unit along the new X axis. This point is at $(-\sin\phi, \cos\phi, 0)$ so the X co-ordinate of P (x, y, z) relative to the new X axis is $y\cos\phi - x\sin\phi$. Similarly to get the new Y co-ordinate we take the scalar product with a point one unit along the new Y axis, &c.

If we had a matrix M with each row the co-ordinates of a unit vector along the respective axis then `M+.×P` would be the full co-ordinates of P relative to the new axes. If we had a matrix N with each **column** a point in an object, then `M+.×N` would be the new set of co-ordinates — the object relative to the new axes. So all we need is the matrix M.

Calling the points one unit along the X, Y and Z axes U_x, U_y and U_z respectively, we have.

$$\begin{aligned} U_x &= (-\sin\phi, \quad \cos\phi, \quad 0) \\ U_y &= (-\cos\theta\cos\phi, \quad -\cos\theta\sin\phi, \quad \sin\theta) \\ U_z &= (\sin\theta\cos\phi, \quad \sin\theta\sin\phi, \quad \cos\theta) \end{aligned}$$

which gives us our matrix.

This gives us the following function which takes a left argument of theta and phi and a right argument of a matrix of points and returns the points relative to the axes defined by theta and phi.

```
 r←angles Transform points;theta;phi;sinT;sinP;cosT;cosP;rotmat
⍝ Return the matrix <points> in the co-ordinate system defined by <angles>
⍝ Each row of <points> represents the x, y, and z co-ordinates of a point
⍝ The new axes are defined as follows :-
⍝  X - in the old XY plane at <phi> anticlockwise from the old Y axis
⍝  Y - normal to the new ZX plane
⍝  Z - at <theta> to old Z with its projection in old XY at <theta> to old X
```

```
 theta phi←angles
⍝ Construct the rotation matrix
 sT sP cT cP←,1 2∘.○theta phi÷180
 rotmat←3 3⍴(-sP)cP 0(-cT×cP)(-cT×sP)sT(sT×cP)(sT×sP)cT

⍝ This is equivalent to ⍉rotmat+.×⍉points
 r←points+.×⍉rotmat
```

So the process of projecting the wire-frame object onto a plane perpendicular to the line from the observer to the origin can be summarised:

- collect the unique set of points of the wire frame
- generate a new set of axes with the observer on the new Z axis
- find the co-ordinates of the points relative to the new axes
- project the points onto the new XY plane
- join the projections of any points that are joined in the old object

The Mechanics of the Game

The code is in three parts. The function `Make` sets up the form, menu, lines and points and puts event handlers on the form for `KeyPress` and `MouseDown`. It also sets up the co-ordinates of the points, the pairs of points joined by lines and the sets of points that signify a win if occupied by the same colour.

```
 Make n;inds
⍝ Set up 3-d Os and Xs game for n×n×n
⍝ Create form with lines and circles and initialise global variables

⍝ Set up globals for point co-ordinates, line indices and win-line indices
 points←(⍉(3⍴n)⊤⍳n*3)-(n-1)÷2
 inds←n n n⍴⍳n*3
 wins←((⍳3)⌽¨⊂⍳3)⍉¨⊂inds
 wins←↑⍪/((⊂3⍴0)⍉¨(1↑wins),⌽¨wins),((⊂0 1 1)⍉¨wins,⌽¨wins),,[0 1]¨wins
 lines←,[0 1](1⌽n↑1 1)/↑⍪/((⍳3)⌽¨⊂⍳3)⍉¨⊂inds

⍝ Set up the state of the game
 colour←(n*3)⍴0
 player←1

⍝ Set up the viewing position
 R theta phi←(3×n)90 180
 ⍝ Make the form and its component parts
 'form'⎕WC'form'('sizeable' 0)('visible' 0)('3d' 'none')
         ('coord' 'pixel')('size'(2⍴¯60+⌊/↑'.'⎕WG'devcaps'))
 'form'⎕WS('xrange'(1-n)(n-1))('yrange'(n-1)(1-n))
 'form'⎕WS'event'('KeyPress' 'Spin')('MouseDown' 'Place')
 'form.lines'⎕WC'poly'(2 2⍴0 0 1 1)('coord' 'user')('fcol'(⊂192 192 192))
 'form.points'⎕WC'circle'(0 0)('fstyle' 0)('coord' 'user')('radius' 0.05)
 'form.menu'⎕WC'menubar'
 'form.menu.game'⎕WC'menu' 'Game'
 'form.menu.game.new'⎕WC'menuitem' 'New'
                         ('event' 'Select' '⍎colour[]←0 ⋄ Draw')

⍝ Draw the current state of the game from the current position
 Draw

⍝ Show the form and pass focus to it
 'form'⎕WS'visible' 1
 ⎕NQ'form' 40
```

The handlers for *KeyPress* and *MouseDown* contain the other two parts of the code. The *KeyPress* handler adjusts the position of the observer and draws on the form the projection of the game relative to that position. The cursor keys increase or decrease *theta* and *phi* while the + and – keys decrease and increase *R* respectively (+ was considered more intuitive for "nearer").

```
 Spin msg;key;shift
⍝ On a KeyPress increment or decrement <R>, <theta> or <phi>
 key shift←¯2↑msg
⍝ <theta> is changed on cursor up and cursor down
⍝ <phi> is changed on cursor left and cursor right
⍝ If Ctrl is pressed the change is 15 degrees
 theta phi←theta phi+(1+14×2=shift)×+/2 2⍴¯1 1 ¯1 1×38 40 37 39=key
⍝ R is decremented by + and incremented by -
 R←3⌈R+0.2×+/¯1 1×375 378=shift+2×key
⍝ Redraw from the new position
 Draw
```

The *MouseDown* handler handles the playing of the game. It compares the mouse position (adjusted for the x and y range of the form) with the projected positions of the free points and determines which one it is near. The criterion of nearness is that when a counter is placed on the point its projection should encompass the mouse position. It then places a counter of the player's colour in *colour* at the corresponding position for that point, changes the player and checks for a win.

```
 Place msg;pos;range;hit
⍝ Place the current player's counter in the place indicated by the mouse
⍝ If the game is already won leave early
 →0/⍨∨/∧/(3-player)=colour[wins]
⍝ Get the mouse position in XY order and adjust to the form's co-ordinate
system
 pos←⌽2↑2↓msg
 range←⊃'form'⎕WG'xrange' 'yrange'
 pos←range[;0]+(-/⌽range)×pos÷⌽'form'⎕WG'size'
⍝ Find the projected points that are near enough to have been hit ...
 hit←((+/2*⍨projected-(⍴projected)⍴pos)<size*2)/⍳↑⍴projected
 →0/⍨0∊⍴hit
⍝ ... and take the first, if it is free
 hit←↑hit
 →0/⍨×colour[hit]
```

```
⍝ Set it to this player and change player
 colour[hit]←player
 player-⍨←3
⍝ Draw the taken points
 DrawPoints
⍝ Detect a win and notify
 →0↓⍨∨/∧/(3-player)=colour[wins]
Win:Win_msg((2-player)⊃'Red' 'Blue'),' has won'
```

Detecting a Win

If any line of *n* points in the cube is all the same colour then the player of that colour has won. The colour of the points is held as a vector of 0s, 1s or 2s with 0 representing an empty point. If reshaped into a cube the vector will map onto the points it represents. To detect a win we need to construct a matrix of *n* columns the rows of which represent all winning lines. Using this to index out of the colour vector we can check to see whether any potentially winning row is all the same colour.

First we make an array of the same shape as the game cube but with as elements the numbers from 0 to n^3-1. The last axis represents our first set of winning lines. If we transpose the array with 1 2 0 and 2 0 1 and take the last axis each time we get the other trivial winning lines. These are the wins parallel to the X, Y and Z axes. We then have two planes of diagonal wins for each transposition. These are given by the 0 1 1 transpose of each array and of its reflection in the last axis. Finally we have the lines connecting the vertices of the cube through its centre. These are given by the 0 0 0 transpose of the array and all its major reflections.

With the 4×4×4 cube this process gives us 76 winning lines of 4 indices. The matrix is set up in the initialisation code and the checking is done each time a counter is placed.

A Word about *R*

The variable *R* represents the notional distance of the observer from the screen, and is in the units of the wire frame's co-ordinate system. In order to get a realistic rendering the best thing to do is to measure the distance from your eye to the screen and divide it by the distance on your screen between two adjacent points in the wire frame. While this gives a realistic 3-d representation, for the game a smaller value of *R* is often better, as it "opens out" the frame allowing more differentiation between front and back points.

This whole treatment also assumes that the line between the observer and the centre of the wire frame is normal to the screen. If it isn't then the eye will detect some distortion. However it is quite surprising how much the human brain will compensate for when it knows that what it is seeing is meant to be cubic.

The Competition

The competition is to write a function that will play the computer's move in the game described above, so that a player can play against the computer. The prize will be free registration for one person at APL96 in Lancaster at the end of July.

The entries to the competition will be played against each other in a league and the top two will play a final game. If at any stage one player has taken more than five minutes longer in total than its opponent (on a P90), it will automatically lose the game. These are the only criteria by which entries will be judged.

Each entry should be a dyadic result-returning function (and perhaps some subfunctions) which will be used in the following way.

Arguments and Result

Left	a numeric scalar 1 or 2, indicating for which player the move should be made
Right	a numeric vector of length *n*⋆3, with elements 0, 1 or 2 If reshaped into a cube each element will correspond to a position on the board, 0 indicating that it is free and 1 or 2 indicating that it is occupied by a piece of the corresponding player.
Result	the origin-0 index of the position in the right argument in which to place the player's piece

The function will be run with a 4×4×4 board under Dyalog APL/W with ⎕*IO* zero and ⎕*ML* three. Both of these system variables can be localised. The migration level is relevant only to second generation features and determines the level of compatibility with APL2, 0 for least and 3 for most.

All entries should be received by the editor on or before 1st March 1996.

Powerful & Easy Graphics; a First Response

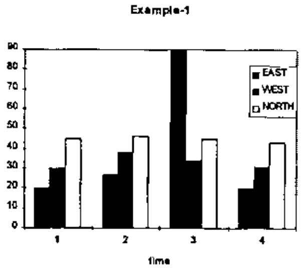
by Adrian Smith (in reply to F.H.D. van Batenburg)

Background

This is mostly in reply to Eke van Batenburg's article *"Powerful and Easy Graphics; a Cry for Help"* [0] which was printed in Quote-Quad Vol.25 No.3. Because many Vector readers will not have seen the original, I will reproduce parts of it here (mostly the figures) as required.

His basic requirement was for a simple (data-driven) language which would allow the APL programmer - in virtually any environment from Mac to Atari to workstation - to generate effective business graphics with the minimum of fuss. At APL95 I ran a 3-hour workshop on the "Rain" language for generating 2-dimensional plots, and at a subsequent panel session on portability, it was generally accepted that this language could form the basis for a portable standard [1].

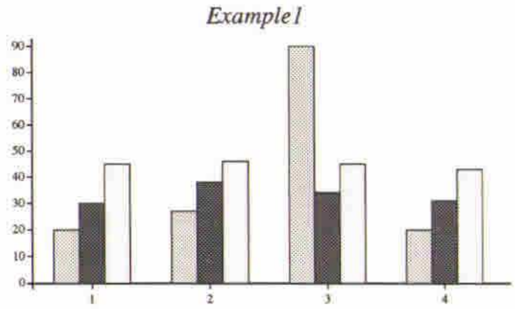
The main difference in approach between Rain and Eke's specification is that he saw the format (as well as the data) being passed as APL data to some kind of *PLOT* function.



```
data[1]←⊂4 4⍴     1 20 30 45 ,
   2 27 38 46 ,
   3 90 34 45 ,
   4 20 31 43

data[2]←⊂'xy'
data[3]←⊂'T%Example-1'  'X%time'
data[4]←⊂ ...
```

... and so on. His data is a 4-element vector, of which the first element is the values to be plotted and the subsequent elements provide the chart style and other possible formatting (titles, keys and so on).

For simple examples (one data-set, one plot at a time) I think this works very nicely, and it would have been very easy to write the Rain functions to work in this way. I chose instead to use a more functional approach mainly because I couldn't see how I could specify more complex plots - such as a line graph superimposed on a bar chart - in a single call. Of course internally Rain simply has a large table of Property-Value pairs which are set by many of the function calls. To produce a barchart similar to Eke's example we would start with something like this:



```
r←Eke1;data
 data←4 3⍴20 30 45,27 38 46,
          90 34 45,20 31 43
 ch.Set('Head' 'Example1')
       ('Xlab' 1 2 3 4)
 ch.Set('Style' 'forcezero')
       ('gap' 0)
 ch.Bar data
 r←ch.Close
```

... where I needed to over-ride the default bar spacing and force both axes through zero to get the same effect. Please note that the syntax used in the examples from Rain is (at the moment) specific to Dyalog APL, as it uses a *ch* namespace to contain all the plotting functions and working variables.

Extending the Syntax to 3D Charts

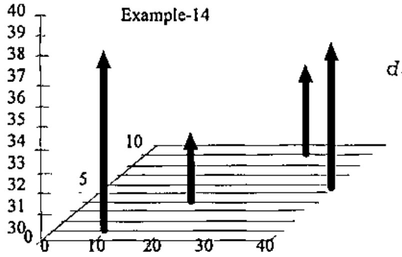
I think I can take it as read that this syntax will work nicely for all the standard 2-dimensional charts. It is easy to add new properties (a radar-style chart may well need some extra specification) as required, as *ch.Set* simply checks against column-1 of a four-column table:

```
      )cs ch
#.ch
      20 4↑Δchdefaults
 Prop  Description      Type  [values]
 ST    Chart Style      ST    /LINE/MARK/WIPE/HALO
 XS    X-axis Style     ST
 YS    Y-axis Style     ST
 ZS    Z-axis Style     ST
 HS    Heading Style    ST
 KS    Key Style        ST
 NS    Note Style       ST
 VS    Value-tag Style  ST    /OPAQUE
 HE    Main Heading     CV
 XC    X-caption        CV
 YC    Y-caption        CV
 ZC    Z-caption        CV
 VA    Value Labels     VV
```

KE	*Legends for Key*	*VV*	
XL	*X-Labels*	*VV*	
YL	*Y-Labels*	*VV*	
ZL	*Z-Labels*	*VV*	
HF	*Heading Font*	*AS*	4 18 3
CF	*Caption Font*	*AS*	6 10 2

... where the type determines the validation, for example *VV* is a vector of text vectors, *AS* is an Axis Specification like *'blue,dash,fine'* (colour, line-style, nib weight) and so on. As long as the new property falls within the existing types (such as the Z-axis labels which are just another sort of axis label) there is very little work to be done in allowing *ch.Set* to accept it. At the moment all the properties must be unique on the first two letters - this dates from Rain's APL*PLUS/PC days when I was anxious to keep the code as compact as possible, and it does make dyadic iota and membership run a touch faster in Dyalog.

Here are some of the 3D chart types that Eke proposes as generally useful; it would be helpful to have some feedback (and ideally some real data) on the kind of data-set that these would best be applied to.



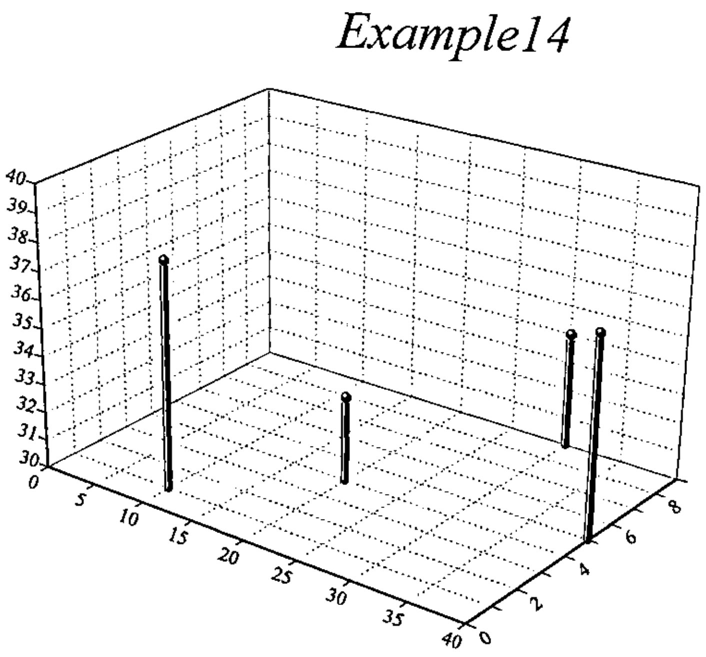
```
data[1]←⊂4 3⍴  10 1 38,
  20 4 33,
  30 9 34,
  40 5 37
```

A good way of starting an argument is to draw three axes on a scrap of paper and ask the people in the room to label them 'X', 'Y' and 'Z'. Eke and I both take the view that Z is always the vertical dimension, and that what you have on the plane is a standard XY plot. This one seems to me to be quite easy to specify, as it is effectively just a scatter plot with risers from the XY plane. You may need grids on any of the XY, XZ or YZ backplanes, and of course there are axis captions to be taken care of.

Here is the code which will be needed to make something quite similar in Rain:

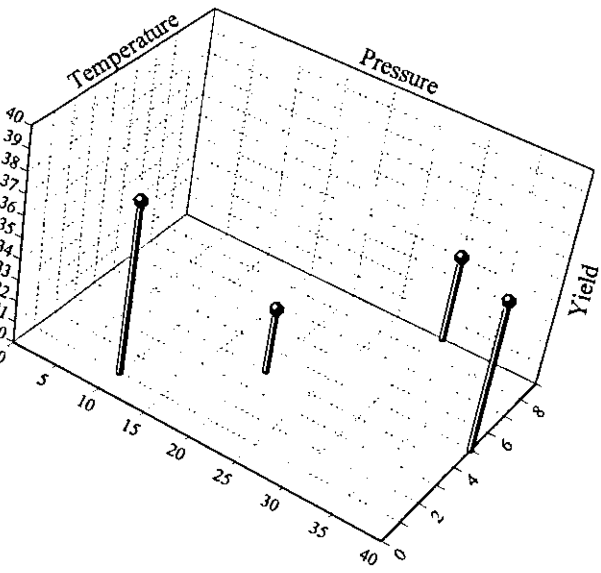
```
r←Eke2;data
 data←4 3⍴10 1 38,20 4 33,30 9 34,40 5 37
 ch.Set('Head' 'Example14')
 ch.Set('zrange' 30 40)('xstyle' 'forcezero')
       ('ystyle' 'forcezero')
 ch.Set('style' 'risers,nolines,grid')('marker' 15)

 ch.Cloud data

 r←ch.Close
```

Marker 15 is a medium Parkhouse Ball [2] which gives a nice solid feel to the data points on the screen; getting a good effect on paper is not so easy:



Placement of the axis labels is a tricky problem. Obviously you can rotate the baseline of a TrueType font, but this is not the same as actually projecting the character shape to the same perspective as the rest of the chart! The effect is rather like inscribing the text along an angled plinth, which I think is acceptable, at least at sensible projections. If we add axis captions and use a rather steeper viewpoint, you can see the sort of effects that result:

Example14 with Captions

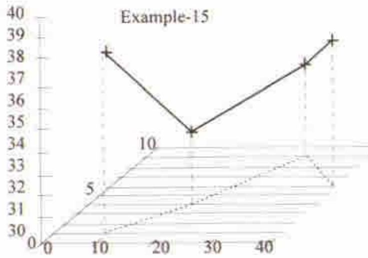


The next interesting question is how the programmer should specify the viewpoint for the chart! In this example it was done with:

```
ch.Set'Viewpoint' 30 25 60
```

which gives *'Roll, Pitch, Yaw'* values in degrees. I then used the perspective calculation from *APL for the Mathematics Classroom* [3] to convert from XYZ triples to XY values with a given amount of perspective distortion. I think I may change this to use a *'theta' 'phi'* approach (see Duncan Pearson's article in this *Vector*) but I find it very hard to guess at 'good' numbers in the abstract. It is also not clear where the visual centre of gravity should be, i.e. through which point the verticals should remain strictly vertical. More experiments needed, I think!

Eke's next example is a variant if the XYZ-scatter with the points joined and the line echoed down on to the XY plane:

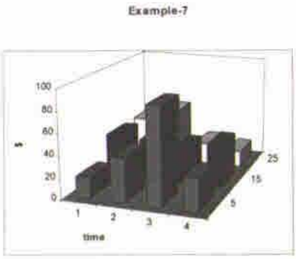


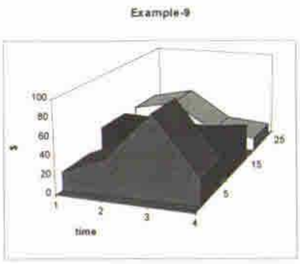
I can also see a valuable use of a simple scatter, with the distribution echoed on all three back planes. This should give an excellent picture of the shape of the relationship between three variables, particularly when this is strongly non-linear in one of the dimensions.

Examples of real data welcome!

More 'Presentational' Charts

Eke's next examples are typical of the 'gee-whiz' charts popularised by Excel and taken to extremes by many of the popular presentation packages. There is the ubiquitous tower chart, and a variety of shaded surface and ribbon styles.





Do we really need these enough to bother, or would it be better to leave the seriously glossy work to packages like CorelChart which major on it? Tower charts are relatively simple to draw, *as long as you restrict the viewpoint to a sensible angle!* All you need are three faces and an edge-trace per tower, working from the back corner, increasing X then decreasing Y as you go. It only gets tricky if the user can look at it from behind, or below, when you need to adjust the drawing order (and draw different faces, lit differently) - my inclination would be to cheat and stay with positive angles and a small perspective distortion, thus eliminating the problem. For tower spacing, I think it would be OK to use the existing barchart *'gap'* property to set the inter-tower spacing in the X direction, and use the *'group-gap'* for the Y-direction.

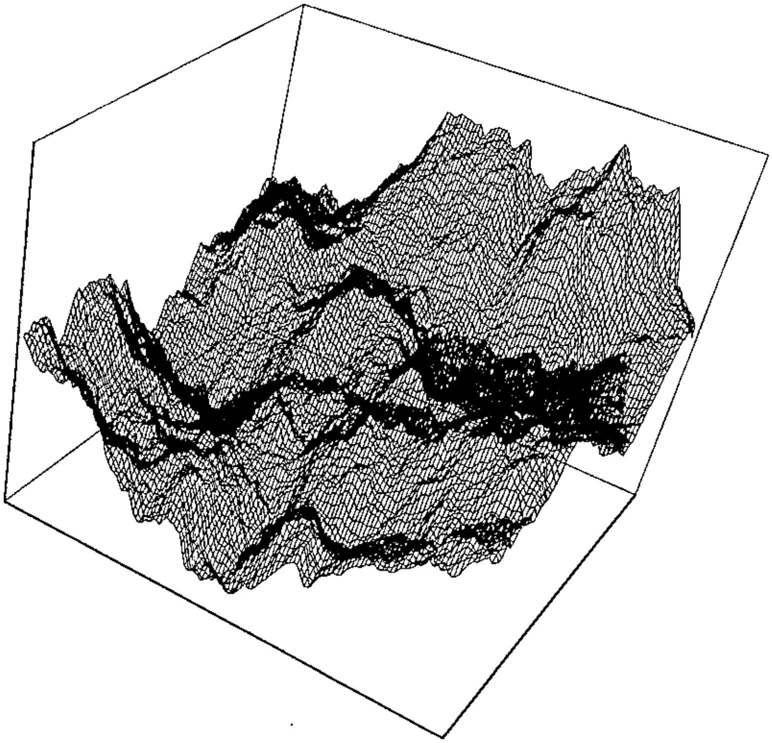
Response Surface Plots

One final type of 3D plot is used by mathematicians to visualise function shape, and also by statistics packages (*Statgraphics* being a good example). In some ways it looks like the XYZ scatter, but the data is presented on a regular XY grid, in a form similar to the data for a tower chart. There is some very pretty code in David Laur's paper [4] for generating fractal terrains, which makes just the right kind of test data:

```
 r←Terrain depth;new;h;v;d;max;lab;ct
⍝ Create and plot fractal terrain to given depth
 max←100 ⋄ m←2 2⍴max÷2

Loop:→lab←1+(depth⍴Loop),End,ct←1
 h←1⌽m ⋄ v←1⊖m
 d←h+v+1⌽v
 new←(0.5×(m+h),m+v),0.25×m+d
 new←(max÷2*ct)Jiggle new
 m←¯1 ¯1↓(2×⍴m)⍴1 3 2⍉(2,1 2×⍴m)[3 1 2]⍴m,new
End:→lab[ct←ct+1]

 'Now to plot it ...'
 ch.Set('style' 'nomarkers')('Head' 'Fractal;Terrain')
       ('hstyle' 'right')
 ch.Set('Xstyle' 'plain')('Ystyle' 'plain')('Zstyle' 'plain')
 ch.Set('Viewpoint' 30 25 60)('hmar' 12)('vmar' 48 12)
 ch.Resp m
 r←ch.Close


 j←max Jiggle n;r;s
⍝ For fractal terrains (see Laur ToT V p.26)
 r←1000 ⋄ s←0.5
 j←n+max×((?(⍴n)⍴r)÷r)-s
```

This algorithm works by dividing an initial 2-by-2 matrix along each edge and at the centre, perturbing the new points by a reducing random amount (inversely proportional to the number of divisions already made) and repeating. To go beyond 8 you need a Pentium and a large workspace, but the effects are very nice.

This (still early prototype) plot just draws the lines and (optionally) markers, rather than a properly tiled surface. At the density generated by `Terrain 8` there are enough lines to give quite effective 'hidden line removal' by default, but I think that a version with a correctly shaded surface will be needed too. Again, you can cheat quite nicely if you restrict yourself to a small range of viewpoints and always light it from the top left.



Fractal Terrain

Making Plots More Interactive

The major limitation of the *Rain* approach to charting is that the plots are effectively dead, once drawn. Eke rightly asks for the possibility of the user interacting directly with the chart, for example to change the heading by clicking on it, or to add text notes in a defined area by marking this with the mouse. He sees this as the result of a function which takes a chart specification as its argument, and returns the modified specification when the user completes his changes and clicks <OK>. This is a very clean approach, and is a major advantage of his completely data-driven style. However I wonder if it is flexible enough to be embedded in any conceivable APL application. For example, the way in which you request a heading might need to be adapted to the data you are plotting (say to offer a pick-list of region codes and expand the chosen item to its full name), or you might simply want to prompt in Finnish.

My suggestion is that there should be a set of defined chart events, which are triggered by the user double-clicking (or using a right-mouse menu) over areas such as the heading, axes, axis captions, keys, and so on. On a 2-dimensional plot it is easy enough to convert back from a drag-selection on the screen into XY co-ordinates, which could then be used by the application to identify any data points in the selected region; obviously on a 3D plot this identification is only possible where X and Y are known in advance, for example on response surfaces.

Fast-moving Data

Eke's final plea is for the possibility of setting up all the framework once, then having the line or bars leap about fast enough to track the progress of a simulation; typically once every second or faster. I am not sure whether it makes sense to do this with the same tool that you would use for a final 'publication-quality' chart. It can also do without a lot of fancy code for auto-ranging axes, as you almost certainly know the range of permissible values in advance, and what's more the early iterations are unlikely to generate a sensible range anyway!

Summary

I hope this adds some useful detail to Eke's original proposal, as well as raising some of the issues associated with an extension of our plotting space to 3 dimensions. *Rain* is shipped as shareware with Dyalog APL, and was demonstrated in prototype form under APL*PLUS III at San Antonio. It will also be shipped with the next release of J for Windows, so the majority of modern APL platforms are or can be covered. It does not (yet) do all the things Eke has asked for - what I need now is some feedback on what is most urgent, otherwise I will gradually add the things that I need, or that are most fun to do. I know that this is how Falkoff and Iverson devised APL in the first place, but it is probably not the best way to go on.

References

[0] F.H.D. van Batenburg, *Powerful and Easy Graphics; a Cry for Help,* Quote-Quad Vol.25 No.3 pp 49-56

[1] *Report on Workshop on Portability,* Vector Vol.12 No.2 page 59

[2] Graham Parkhouse, *Graphics from First Principles,* Vector Vol.7 No.4 pp 83-98

[3] Norman Thomson, *APL for the Mathematics Classroom,* Springer-Verlag 1989

[4] David M. Laur, *A Fractal Experimenter's Toolkit,* APL Tools of Thought V, New York, 1987, pp 21-37

THE RANDOM VECTOR

A Note on Programming Style

by Roger K.W. Hui and Kenneth E. Iverson

We were pleased to see the uses of J in the October *Education Vector*, particularly the Muller, van Woudenberg, and Young program for divided differences and the fact that that it resulted from a mathematics program at Strathclyde. We would like to use it together with contributions from Norman Thomson in the same issue to discuss certain matters of programming style: although MWY followed "the style of a Pascal program, which is a natural way ...", the use of vectors and matrices would be more natural to mathematicians, and would better lend itself to analysis.

Divided Differences

As shown by the use of control structures in Thomson's *J-ottings 7*, the MWY program can be made even more Pascal-like as follows:

```
dd0 =: 4 : 0
 i=.0
 n=.#x.
 z=.(i<.n){.d=.y.
 while. n>i=.1+i do.
  dx=.(-i)}.(i|.x.)-x.
  dy=.2 --/\ d
  d =.dy%dx
  z =.z,{.d
 end.
)

   x=.0 1 5 8
   y=.4 6 18 6
   x dd0 y
4 2 0.2 _0.15
```

The expression for dx computes differences in x that are i apart, by taking the difference between rotating x by i and x itself and discarding the last i entries. The expression for dy divides d into overlapping windows of size 2 and applies --/ (insert subtract from) to each window.

The computation can also be expressed tacitly, an expression that facilitates investigation of the internal workings of the algorithm:

```
dix=: -@[ }. |. - ]
dx =: >:@-&# dix [
dy =: 2: -~/\ ]
dd =: (dy % dx)^:(i.@#@])

   x dd y              {."1 x dd y
    4  6 18 6  4 2 0.2 _0.15
    2  3 _4 0
  0.2 _1  0 0
_0.15  0  0 0

   x dx y              1 dix x
1 4 3                    1 4 3

   x dy y              2 dix x
2 12 _12       5 7

                       3 dix x
               8

   f=.dy % dx

   x f y
2 3 _4

   x f x f y
0.2 _1

   x f x f x f y
_0.15

   x f^:3 y
_0.15

   x f^:0 1 2 3 y
    4  6 18 6
    2  3 _4 0
  0.2 _1  0 0
_0.15  0  0 0
```

Choleski Decomposition

MWY and Thomson presented programs for the Choleski decomposition. We present here a recursive version. Given a positive definite matrix A, the Choleski method computes a lower triangular matrix L such that `A-:L x h L`, where `x=:+/ .*` is the matrix product and `h=:+@|:` is the conjugate transpose. A recursive solution reveals itself by considering A as a 2-by-2 matrix of matrices and substituting appropriate matrix operations for scalar ones in the Choleski method for a 2-by-2 matrix of scalars.

```
x=: +/ . *                        matrix product
h=: +@|:                          conjugate transpose

Choleski =: 3 : 0
 n=.#A=.y.
 if. 1>:n do.
  13!:8(,(A=|A)>0=A)}.12          check for positive definite
  %:A
 else.
  p=.>.n%2 [ q=.<.n%2
  X=.(p,p){.A [ Y=.(p,-q){.A [ Z=.(-q,q){.A
  L0=.Choleski X
  L1=.Choleski Z-(T=.(h Y) x %.X) x Y
  L0,(T x L0),.L1
 end.
)

   A                              A random positive definite matrix
  33    7j_8 7j_10     3j_4
 7j8      28   2j4 _10j_11
7j10    2j_4    22      3j3
 3j4 _10j11  3j_3       16

   L=. Choleski A

   $L
4 4

   A -: L x h L                   A true decomposition
1

   *./,(>:/~i.#L) >: 0~:L         L is lower triangular
1

   [ B=. 3 3$ 1 4 6  4 0 3  6 3 2
1 4 6
4 0 3
6 3 2
```

```
   Choleski B                   B is not positive definite
|assertion failure
|       13!:8(,(A=|A)>0=A)}.12
```

QR Decomposition

Thomson's *Technical Note on Matrix Decomposition* presented the QR and LU decompositions. Both are amenable to the recursive approach used above for the Choleski decomposition, with array operations playing a key role. Here, we present a solution for the QR decomposition; a recursive array LU decomposition can be found in Aho, Hopcroft, and Ullman [1974, Section 6.4].

Given a matrix A where `>:/$A`, the QR decomposition produces an orthogonal matrix Q and a square upper triangular matrix R such that `A-:Q x R`. A recursive solution reveals itself by considering A as a 2-column matrix of matrices and substituting matrix operations for scalar and vector ones in the Gram-Schmidt method for two vectors.

```
x=: +/ . *                     matrix product
h=: +@|:                       conjugate transpose

QR =: 3 : 0
 n=.{:$A=.y.
 if. 1>:n do.
  (A%{.(,norm),0);norm=.%:(h A) x A
 else.
  m =.>.n%2
  A0=.m{."1 A
  A1=.m}."1 A
  ('Q0';'R0')=.QR A0
  ('Q1';'R1')=.QR A1 - Q0 x T=.(h Q0) x A1
  (Q0,.Q1);(R0,.T),(-n){."1 R1
 end.
)

   [ A=. j./_8+?2 7 4$20       A random matrix
 _6j6  7j10  1j7 2j_3
_4j_8  _8j6 5j_2  5j4
 10j7 _1j11 2j_1 8j_4
_8j11  _7j6  2j7  5j5
_8j_7  _1j4 _7j9 0j_3
    5   3j7 10j1 8j_4
 2j_3 _7j_1 5j_5  0j1

   $QR A                       Result is a 2-element boxed vector
2
```

```
   'QR'=.QR A            Assign first box to Q, second to R

   $Q
7 4

   $R
4 4

   A -: Q x R                      A true decomposition
1

   >./,|(=i.{:$Q)-(h Q) x Q        Q is orthogonal
8.51083e_16

   *./,(<:/~i.#R) >: 0~:R          R is upper triangular
1
```

References

[1] Aho, Alfred V., John Hopcroft, and Jeffrey D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1974.

[2] Muller, Antje, Tineke van Woudenberg, and Alistair Young, *Two Numerical Algorithms in J*, The Education Vector, Volume 12, Number 2, 1995, pp 26-30.

[3] Strang, Gilbert, *Linear Algebra and its Applications*, Academic Press, 1976.

[4] Thomson, Norman, *J-ottings 7*, The Education Vector, Volume 12, Number 2, 1995, pp 21-25

[5] Thomson, Norman, *Technical Note on Matrix Decomposition*, The Education Vector, Volume 12, Number 2, 1995, pp 31-32.

The Incomplete Elliptic Integrals and APL: an Extension

Joseph De Kerf

Elliptic integrals of the first kind $K(\phi;p)$ and of the second kind $E(\phi;p)$ may be defined as:

$$K(\phi;p) = \int_0^{\phi} \frac{d\theta}{\sqrt{1-p^2\sin^2\theta}}$$

$$E(\phi;p) = \int_0^{\phi} \sqrt{1-p^2\sin^2\theta} \;\; d\theta$$

with $p^2 \le 1$ or $-1 \le p \le 1$.

For $\phi = \Pi/2$ those integrals are defined as the complete elliptic integrals of the first kind $K(p) = K(\Pi/2;p)$ and of the second kind $E(p) = E(\Pi/2;p)$. For $0 \le \phi < \Pi/2$ the elliptic integrals of the first kind $K(\phi;p)$ and of the second kind $E(\phi;p)$ are called the incomplete elliptic integrals, the complete elliptic integrals being their boundary values. It should be noted that:

$K(-\phi;p) = -K(\phi;p)$ and $E(-\phi;p) = -E(\phi;p)$.

With all this in mind, APL user-defined functions for calculating those elliptic integrals for $-\Pi/2 \le \phi \le \Pi/2$, based on the common mean algorithm [1], were given in [2]: IEI1 for calculating the elliptic integrals of the first kind $K(\phi;p)$ and IEI2 for the elliptic integrals of the second kind $E(\phi;p)$ (*cf.* Figure 1.)

```
        ∇IEI1[⎕]∇
       ∇R←F IEI1 P;I;A;B
[1]     →((1≠|P)∨(○÷2)=|F)/LAB1
[2]     →0,R←⍟3○0.5×F+○0.5
[3]    LAB1:R←(2+I←¯1),(1-P*2)*0.5
[4]    LAB2:A←(-/R)×1○F×2*1+I←I+1
[5]     B←2×+/R×(2 1○F×2*I)*2
[6]     F←F-(¯3○A÷B)÷2*I+1
[7]     →(≠/R←(0.5×+/R),(×/R)*0.5)/LAB2
[8]     R←F÷0.5×+/R
       ∇
```

```
      ∇IEI2[⎕]∇
     ∇R←F IEI2 P;I;S;T;A;B
[1]   →(1=|P)/0,R←1○F
[2]   R←(1+S←T←1+I←¯1),(1-P*2)*0.5
[3]  LAB:A←(-/R)×1○F×2*1+I←I+1
[4]   B←2×+/R×(2 1○F×2*I)*2
[5]   F←F-(¯3○A÷B)÷2*I+1
[6]   S←S+(-/R*2)×2*I
[7]   T←T+(-/R)×1○F×2*I+1
[8]   →(≠/R←(0.5×+/R),(×/R)*0.5)/LAB
[9]   R←(T÷2)+(2-S)×F÷+/R
     ∇
```

Figure 1

In some applications, however, we need the elliptic integrals of the first kind $K(\phi;p)$ and of the second kind $E(\phi;p)$ for values of the argument $|\phi| > \Pi/2$. For $p \neq \pm 1$ and $|\phi| > \Pi/2$ the user-defined functions shown still give the correct results, but for $p = \pm 1$ and $|\phi| > \Pi/2$ they may fail:

For $p = \pm 1$ and $|\phi| \geq \Pi/2$, the elliptic integral of the first kind $K(\phi;p)$ is always infinite (∞), while the user-defined function IEI1 eventually returns a value instead of a domain error report, for example when $3\Pi/2 \leq \phi < 5\Pi/2$.

For $p = \pm 1$, the user-defined function IEI2 returns $\sin\phi$, but this is only correct for $-\Pi/2 \leq \phi \leq \Pi/2$. Indeed:

for $\Pi/2 \leq \phi \leq 3\Pi/2$	$E(\phi;p) = 2 + \sin(\phi - \Pi)$
$3\Pi/2 \leq \phi \leq 5\Pi/2$	$= 4 + \sin(\phi - 2\Pi)$
$5\Pi/2 \leq \phi \leq 7\Pi/2$	$= 6 + \sin(\phi - 3\Pi)$
....	....

In general: $E(\phi;p) = 2n + \sin(\phi - n\Pi)$ where n is the result of rounding off to an integer the quotient ϕ/Π. It should be noted that this procedure is also valid for $\phi \leq -\Pi/2$.

APL user-defined functions, IEI1 for the elliptic integrals of the first kind $K(\phi;p)$ and IEI2 for the elliptic integrals of the second kind $E(\phi;p)$, valid for every value of the argument ϕ are given in Figure 2.

```
      ∇IEI1[⎕]∇
     ∇R←F IEI1 P;I;A;B
[1]   →((1≠|P)∨(○÷2)≤|F)/LAB1
[2]   →0,R←⍟3○0.5×F+○0.5
[3]  LAB1:R←(2+I←¯1),÷÷(1-P*2)*0.5
[4]  LAB2:A←(-/R)×1○F×2*1+I←I+1
[5]   B←2×+/R×(2 1○F×2*I)*2
[6]   F←F-(¯3○A÷B)÷2*I+1
[7]   →(≠/R←(0.5×+/R),(×/R)*0.5)/LAB2
[8]   R←F÷0.5×+/R
     ∇

      ∇IEI2[⎕]∇
     ∇R←F IEI2 P;N;I;S;T;A;B
[1]   →(1=|P)/0,R←(2×N)+1○F-○N←⌊0.5+F÷○1
[2]   R←(1+S←T←1+I←¯1),(1-P*2)*0.5
[3]  LAB:A←(-/R)×1○F×2*1+I←I+1
[4]   B←2×+/R×(2 1○F×2*I)*2
[5]   F←F-(¯3○A÷B)÷2*I+1
[6]   S←S+(-/R*2)×2*I
[7]   T←T+(-/R)×1○F×2*I+1
[8]   →(≠/R←(0.5×+/R),(×/R)*0.5)/LAB
[9]   R←(T÷2)+(2-S)×F÷+/R
     ∇
```

Figure 2

For $p \neq \pm 1$ and/or $|\phi| \geq \Pi/2$, the flow of user-defined function IEI1 is branched from line 1 to line 3. However, in line 3 the expression `(1-P*2)*0.5` has been substituted by the equivalent expression `÷÷(1-P*2)*0.5`. This means that for $p = \pm 1$ and $|\phi| \geq \Pi/2$, a domain error report is returned.

In line 1 of the user-defined function IEI2, the result R is *a priori* set to $2n + \sin(\phi - n\Pi)$, with n assigned the value as defined above. If $p = \pm 1$, flow of execution returns to the invoking expression and $2n + \sin(\phi - n\Pi)$ is returned. If $p \neq \pm 1$, flow of execution continues in line 2 and the *a priori* set result is replaced by its actual value.

References

[1] J. De Kerf, *The Common Mean and APL,* Vector 11.3, pp 21–23, 1995

[2] J. De Kerf, *The Incomplete Elliptic Integrals and APL,* Vector 12.2, pp 95–100, 1995

Joseph De Kerf
Rooienberg 72
B-2570 Duffel
Belgium

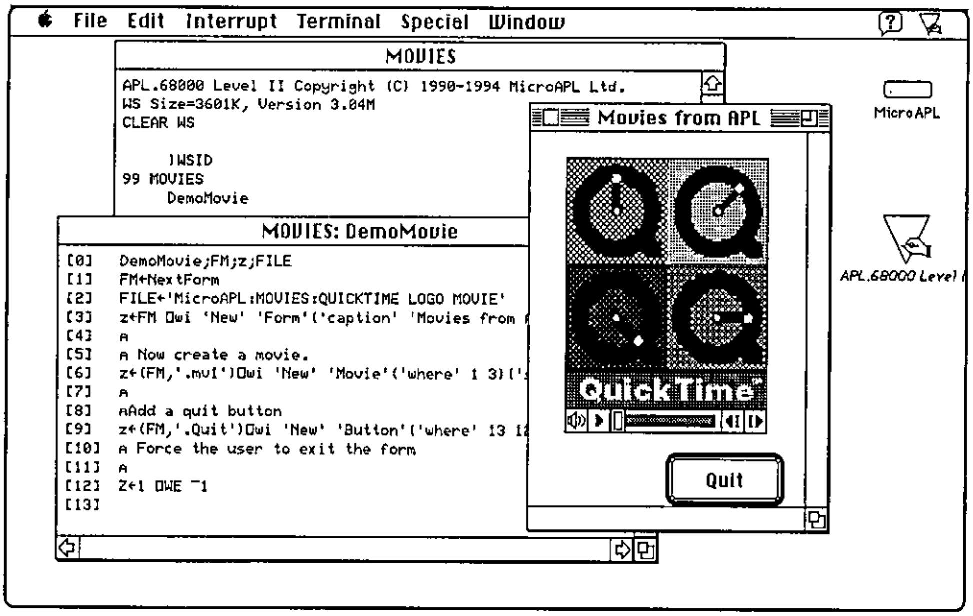
TECHNICAL SECTION

This section of VECTOR is aimed principally at those of our readers who already know APL. It will contain items to interest people with differing degrees of fluency in APL.

Contents

APL for the Apple Macintosh

Since 1985 MicroAPL has pioneered the use of APL in graphical environments. Our latest version of APL.68000 Level II for the Apple Macintosh is now available, offering dramatically enhanced GUI programming facilities.



```
APL.68000 Level II Copyright (C) 1990-1994 MicroAPL Ltd.
WS Size=3601K, Version 3.04M
CLEAR WS

      )WSID
99 MOVIES
      DemoMovie
```

```
[0]   DemoMovie;FM;z;FILE
[1]   FM←NextForm
[2]   FILE←'MicroAPL:MOVIES:QUICKTIME LOGO MOVIE'
[3]   z←FM ⎕wi 'New' 'Form'('caption' 'Movies from
[4]   ⍝
[5]   ⍝ Now create a movie.
[6]   z←(FM,'.mv1')⎕wi 'New' 'Movie'('where' 1 3)('
[7]   ⍝
[8]   ⍝Add a quit button
[9]   z←(FM,'.Quit')⎕wi 'New' 'Button'('where' 13 1
[10]  ⍝ Force the user to exit the form
[11]  ⍝
[12]  Z←1 ⎕WE ¯1
[13]
```

APL.68000 Level II for the Macintosh includes the folowing features:

- Runs on all models of Apple Macintosh
- Native version for the Power Macintosh
- Conforms closely to the APL2 specification
- Uses the standard Mac interface
- Object-based GUI programming via ⎕WI
- Full event handling via APL callbacks
- Free runtime version with application packager

Accelerated for Power Macintosh

MICRO APL

MicroAPL Limited	South Bank Technopark, 90 London Road, London, SE1 6LN, UK
Voice:	0171 922 8866
Fax:	0171 928 1006
Applelink:	microapl
Internet:	microapl@microapl.demon.co.uk

Hackers' Corner: Oh ***! I hadn't saved that!

by Dave Phillips

Don't you just love Windows? There you are minding your own business doing a bit of quiet experimentation into how to get things to happen in your *favourite* operating environment when all of a sudden... 'Now where did that go ?!!' ...usually accompanied with a much longer stream of unprintables.

And because you hadn't been too sure where your experiments were leading you hadn't troubled to save the workspace until you'd got something worth saving... ...and now you've got nothing!!

Being a 'bear of very little brain' this happened to me more than once and I was getting tired of it! Something had to be done! An autosave facility seemed to be a necessity.

So, how could I get my APL*Plus III system to *)SAVE* every five minutes, say. If I could have a Windows Timer control around with the requisite 5 minute interval then when the interval popped off perhaps I could have that save my workspace. But I certainly wouldn't want to have to precondition each and every development workspace with a lot of code just for autosave.

Now, in Vector 11.1 Eric Lescasse (p112 et seq) showed how to build a small 'Time' application. More importantly he very clearly demonstrated the independence, in APL*Plus III, of objects from any workspace. He)CLEARed the workspace and showed that the form still functioned properly. This is entirely true with one caveat, all the callback code for the object has to be contained within the definition of the object itself. That is fine when the callbacks are simple APL statements but what if they are of greater complexity requiring one or more functions to be called — if those functions have to be in a workspace 'BANG!!' goes our workspace independence.

Fortunately APL*Plus III objects have the great facility of being able to store data on them — not just 'text' to go in an editbox or a 'list' for a listbox but any old data can be stored in the 'data' property of any form or control — and where goes data so too can go the canonical representation of a function.

I was nearly there; some code to build the form and load the data property with the *⎕VR* of some more code to oversee *⎕SAVE*ing the current workspace; an `onTimer` event to retrieve the function definition and run it; a switch to turn autosaving on and off; and finally cause all this to happen automatically by calling it in `⎕lx` of a workspace which would be loaded as the initial workspace every time I started APL*Plus III.

The form, once created, would be 'owned' by the APL session so there'd be no need to worry about any closedown procedures — closing the session would close and delete all owned forms. Also, since `'New' 'Form'` would produce a form which was open (although in this case it would also be hidden), the Timer control would be active as soon as it was created. There would be no need for a wait-loop on this form, instead it would be activated when Windows generated the timer event. So there I had it!

Of course Windows had to have the last laugh! The Windows SetTimer call has a maximum interval setting of Ushort (i.e. 65536) milliseconds! ...just over 65 seconds. So, a little more work would have to be done by the autosaver unless I was going to be saving every minute! ...and I thought that was probably a little excessive. Also, I thought it would be best to save the workspace as 'AUTOSAVE' rather than have the possibility of a rubbishy experimental state overwrite something I would have prefered to keep intact.

So the final ingredients for this recipe are:

1. An entry in APLW.INI

   ```
   [Config]
   Initial Workspace=D:\Work\aplw1_2\initws
   ```

   to load my initial workspace INITWS.W3

2. INITWS.W3 has

   ```
   ⎕lx ← 'AS_Init'
   ```

3. The On/Off switch for autosaving is simply the existence or not in the current workspace of a global variable '`⍙autosave`' which can be created or erased by setting up a pair of Tools in the APL*Plus III Tools menu option.

   Autosave on

   ```
   ⍙autosave ← 'on'
   ```

   Autosave off

   ```
   0 ρ ⎕ex '⍙autosave'
   ```

   ...and, of course, the bits you've all been waiting for ('WAKE UP AT THE BACK!!')

The Functions to do it ...

```
    ∇ AS_Init;rc
⍝∇ AS_Init - called in ⎕lx
⍝ This creates the hidden Autosave Object which has
⍝ its own callback code entirely self-contained.
⍝ As a consequence this Object is workspace independent.

:if ~(⊂'autosave') ∊ '#' ⎕wi 'children'
            ⍝ if 'autosave' object does not yet exist
    rc← 'autosave' ⎕wi 'New' 'Form' 'Hide' ('data' 0)
            ⍝ build a hidden form with data value 0
    rc← 'autosave.l' ⎕wi 'New' 'Label' ('data' AS_Timer_vr)
            ⍝ apply a label to the form with data set..
            ⍝ ..to the ⎕vr of our onTimer callback
    rc← 'autosave.t' ⎕wi 'New' 'Timer' ('interval' 60000)
            ⍝ apply a timer with a one minute interval
            ⍝ n.b. 'SetTimer' is a Win16 call in Windows 3.1;
            ⍝ and interval is U, therefore max. value is 65536
            ⍝ i.e. approx 1min 5secs (timer is in msec)
    rc ⎕wi 'onTimer' '⍎⎕def ''autosave.l''⎕wi''data'''
            ⍝ when timer interval pops off, ⎕def the data..
            ⍝ ..from the label and execute it
:end
    ∇
```

AS_Timer_vr ← ⎕VR 'AS_Timer' where..

```
    ∇ AS_Timer;m;dt
⍝∇ AS_Timer - callback to handle the Autosave onTimer event
 m←⎕wi ':data' ⋄ m← m+1      ⍝ Recall the current 'minute' from
                             ⍝ the form and increment it.
:if 5 ≤ m                    ⍝ If we've reached 5 'minutes'
    m← 0                     ⍝ ..reset counter to 0
    :if ×⎕nc '⍙autosave'     ⍝ } Is Autosave....
    :andif ⍙autosave≡'on'    ⍝ } ...'switched on'
      ⍙wsid← ⎕wsid           ⍝ if so save real wsid as global
      ⎕wsid← 'AUTOSAVE'      ⍝ ..rename the workspace
      'RESET' ⎕save ⎕wsid    ⍝ ..and save it
      ⎕wsid← ⍙wsid           ⍝ ..and re-establish proper wsid
      dt←,'G<Z9/99/99 99:99>' ⎕FMT 100⊥100|⎕TS[⎕io+2 1 0 3 4]
      ⎕←'AUTOSAVE at ',dt    ⍝ Display an Autosave message
    :end
:end
 ⎕wi ':data' m               ⍝ Store the new 'minute' counter
 0 0 ρ ⎕ex 'AS_Timer'        ⍝ ..and dispose of this code
    ∇
```

...now all I have to do is to remember to turn on the Autosa.......... Oh ****!

TECHNICAL CORRESPONDENCE

Technical Note: Fractions as Sequences of Integers

From: Norman Thomson December 1995

The first problem considered here is that of converting decimal fractions to fractions F in base n. Here are two solutions; which to use depends on whether you are interested in the number of digits in the results, or in the closeness of the approximation. The inverse in each case is given by `(÷n)⊥⌽ 0,F`.

```
z ← n fdiv f;t
⍝ z is fraction f expressed as n-ary fraction
⍝  n has form of left argument of encode
 z ← ⍳0 ⋄ →(0=⍴n)/0
 z ← (⌊t),(1↓n)f÷1|t ← f×↑n

    tol ← 1E¯12
    +F ← (5⍴5)fdiv 0.67296
3 1 4 0 2
    (÷5)decode rotate 0,F
0.67264

    +F ← (9 ⍴ 5)fdiv 0.67296
3 1 4 0 2 4 4 4 4
    (÷5)⊥⌽0,F
0.672959488

z ← n fdiv1 f;t;u
⍝ z is fraction f expressed as n-ary fraction
⍝  algorithm stops when tolerance has been achieved
 z ← ⌊ f ÷ u ← ÷ n
L1:→(tol > t ← u|f)/0
 z ← ⌊ t ÷ u ← u ÷ n ⋄ f ← u|t ⋄ →L1

    +F ← (5⍴5)fdiv1 0.67296
3 1 4 0 3
```

```
      (÷5)⊥⌽0,F
  0.67296
      4 fdiv1 ÷ 3
  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

A related problem is that of expressing a fraction as a sum of best-fitting unit fractions, the algorithm stopping when a given tolerance has been achieved. This is sometimes called Sylvester's algorithm, and is coded:

```
  [0] z ← frac f;t
  [1] ⍝ z is fraction f expressed as sum of unit fractions
  [2]  z ← ⌈ ÷ f
  [3] L1:→(tol > t ← f- ÷ ¯1↑z)/0
  [4]  z ← z,⌈÷f ← t ⋄ → L1

      frac 0.67296
  2 6 159 248438 1.234441339E11
```

Like the first algorithms above, this provides a way in which a fraction can thus be expressed as a sequence of integers. In particular the sequence corresponding to the fractional part of *pi* is

```
       frac 1 |○1
  8 61 5020 12854157
```

Miaou = Cat!

From: Claude Henriod December 1995

Teachers have known for years that from earliest youth one must learn the right word. One calls a cat, a cat.

It ought to be the same with languages in information technology, but it isn't always so. I want to give a simple example drawn from *Vector* Vol 12 No 1 page 88.

The vector *Q* is formed of three numbers:

```
  Q←⎕AI[2] ⋄ <SOME CODE> ⋄ Q←Q,⎕AI[2] ⋄ <SOME CODE> ⋄ Q←Q,⎕AI[2]
```

In the setting of my example I'm not debating the choice of assignation but what follows:

```
  Q[2]-Q[1] ⋄ Q[3] - Q[2]
```

This is the difference between two scalars. Basic, Fortran, C and other languages are capable of doing this sum.

On the other hand APL can do better: the difference between two items from the same vector:

```
    -/ Q[2 1]          or, to be complete,
    -/ Q[2 2 ⍴ 2 1 3]
```

APL2 from IBM (and other APLs) can do these two sums with a simple operator:

```
    ¯2 -/Q
```

THINK. It takes hardly any effort to think of *Q* as a set of values as distinct from some scalars in a variable. Even in as simple an example as presented here, the latter will never be programmed well in APL.

*I deplore the logic of APL*PLUS III which introduced structure, because it makes it easy to program treating variables as scalars, as in Fortran.*

P.S. I leave you the surprise of comparing execution times on your machine with your interpreter. (On my 66MHz PS2 with IBM's APL2 the vector version takes a quarter of the time.)

At Play With J
Heron's Rule & Integer-Area Triangles

by Eugene McDonnell

Preliminaries

This note makes use of several less-well-known parts of J: the fix (`f.`) and Taylor series coefficient (`t.`) adverbs and the polynomial rootfinder verb (`p.`).

To make the following accessible to all readers, the following verbs are defined:

```
C =. @  NB. compose       f C g x <-> f g x
D =. %  NB. divide        18 D 3  <-> 6
H =. -: NB. halve         H 12    <-> 6
I =. ]  NB. identity      I 6     <-> 6
P =. */ NB. product       P 1 2 3 <-> 6
R =. %: NB. square root   R 36    <-> 6
S =. +/ NB. sum           S 1 2 3 <-> 6
Z =. 0: NB. zero          Z 1 2 3 <-> 0
```

The following convention applies to verbs `f`, `g`, and `h`:

```
   (f g h) y <-> (f y) g (h y) NB. (%: , *:) 16 <-> 4 256
```

Heron's Rule

Heron's rule for the area A of a triangle with sides *a*, *b*, and *c* is usually written in two steps. First the semi-perimeter *s* is computed:

```
  s =. (a + b + c) D 2
```

For example:

```
   (13 + 14 + 15) D 2
         42       D 2
21
```

And then the following expression for the area is computed:

```
   A =. R (s * (s - a) * (s - b) * (s - c))
```

Continuing the example:

```
   R (21 * (21 - 13) * (21 - 14) * (21 - 15))
   R (21 *     8     *     7     *     6    )
   R 7056
84
```

Heron's is a scalar-oriented formula, with the lengths of the three sides and the semi-perimeter playing separate roles in the formulation. We make a first approach to an array formulation by considering the triangle to be defined by a 3-item list of side lengths. We then determine the semi-perimeter by a verb `SP`:

```
   SP =. H C S
```

The next step is to replace the three explicit subtractions by appending a zero to the list and subtracting the resulting four values from the semi-perimeter, then taking the product over this result, and finally the square root of the product.

```
   Heron =. R C (P C (SP - (Z , I)))
```

This is a slightly more efficient form than APL expression 318 in the FinnAPL Idiom Library.

```
   Heron 13 14 15
84
```

Fixing the definition of Heron, and giving this fixed version the name `Hrn`, by using the fix adverb (`f.`) yields a form in which the names of defined items are replaced by their values. Doing this insures that changes in the items defined do not alter the definition of the item in which they are used. As a side effect, a fixed verb is generally faster than an unfixed equivalent.

```
   Hrn =. Heron f.
   Hrn
%:@(*/@(-:@(+/) - 0: , ]))
   Hrn 13 14 15
84
```

Integer Heron

In preparing examples for Heron's formula, I thought it would make the examples clearer if I could find triangles having integer sides that also had integer areas. I explored consecutive triplets of integers among the first 200 sets of triplets.

The first step was to build the table of triplets (the table has 200 rows):

```
   T =. 3 ]\ i. 202
   $ T
200 3
```

Its first and last four rows are:

```
   4 _4 {."0 _ T
  0   1   2
  1   2   3
  2   3   4
  3   4   5
196 197 198
197 198 199
198 199 200
199 200 201
```

Applying `Hrn` to the rows of this table gives a list of areas.

```
   Areas =. Hrn"1 T
```

We determine which of the areas are integral (those equal to their own floor):

```
   Mask =. (= <.) Areas
```

And use the mask so found to give us the winning rows of T:

```
   Winners =. Mask # T
   Winners
  1   2   3
  3   4   5
 13  14  15
 51  52  53
193 194 195
```

The areas corresponding to these are:

```
   Hrn"1 Winners
0 6 84 1170 16296
```

The triangle 1 2 3 is degenerate (ugh!).

A Recursive Formula

I looked at Winners for some clue as to how the series could be prolonged, but without success. Then I thought of N. J. A. Sloane's book *A Handbook of Integer Sequences.* I looked in it for the series 1 3 13 51 193 without avail. Then, knowing that each series in the book began with 1, which was sometimes prefixed to a series which began naturally with some other integer, I looked for 1 2 4 14 52 194 and 1 3 5 15 53 195, but again to no avail. Finally, I divided the even column 2 4 14 52 194 by 2, looked for 1 2 7 26 97, and this time struck pay dirt. It was Sloane's sequence 700.

Sloane's entry for series 700 not only gave a number of additional values but, more importantly, it gave a doubly recursive formula for finding the values, in common mathematical notation:

$$A(n) = 4A(n-1) - A(n-2)$$

So now I was able to extend the series as far as I wanted. I wrote a J version of A:

```
A=.  ((4:*A@<:)-(A@<:@<:))`>:@.(<&2)
```

which, as you can see, is doubly recursive in A. It tests whether the argument is less than 2 (`<&2`), giving one plus the argument (`>:`) as result in these cases, and otherwise yields the difference between A of n-1 (`<:`) and A of n-2 (`<:<:`).

I calculated additional results of A, for arguments 6 through 14, derived triplets from the results, and applied `Hrn` to the triplets, and in each case found an integer area. But was I satisfied? No.

Generating Functions

O them doddhunters and allanights, aabs and baas for agnomes, yees and zees for incognits, bate him up jerrybly! *James Joyce, Finnegans Wake, p. 283*

The reason the story carries on is that I was unhappy with the long execution times required by the deeply rooted calling trees of the double recursion. I had to terminate the execution of `A 30` after five hours with no result. My mind turned to the subject of generating functions, something I had often heard about and often, with little or no success, had tried to master. I was stimulated to do this because of three books. These were K. E. Iverson's new book *Concrete Math Companion*; the Ronald Graham, Donald E. Knuth, and Oren Patashnik book *Concrete Mathematics,* which I shall refer to as GKP; and most of all, the H. S. Hall and S. R. Knight book *Higher Algebra,* first edition 1887, and usually referred to as Hall & Knight, worthy successor to Todhunter's *Algebra for Schools and Colleges.*

Both of these books are celebrated by James Joyce in the mathematics chapter of his Finnegans Wake. Iverson's new book and GKP focus sharply on generating functions. From GKP I learned a four-step process that promised to allow me to have my will with arbitrary generating functions. I plodded through their examples, and tried to duplicate their results on my problem. No luck. I turned to Iverson's book, and found out one important thing that GKP had neglected to tell me, that is, that the key to generating functions was the ability to generate the coefficients of Taylor series, something that J is well suited for, since it contains a primitive (`t.`) to do just that. However, that is about all I was able to learn there. Lastly, I got out my rusty red copy of Hall & Knight, and it came through. The examples they gave were of the same kind as mine, that is, they dealt primarily with doubly recursive functions, where the nth term is some linear combination of the two preceding terms. Their explanations were carefully laid out in great detail.

Here is how they go about it.

Given a sequence with a sufficient number of terms, it is possible to describe how to extend the sequence arbitrarily. The first thing to do is to get rid of the notion that we are dealing with a mere list of numbers. Instead we think of the list as being the coefficients in a polynomial with a never-ending set of terms, that is, an infinite series. Thus the list:

```
  1 2 7 26 97 ...
```

in fact defines the first several coefficients of the infinite series:

```
    (1*y^0) + (2*y^1) + (7*y^2) + (26*y^3) + (97*y^4) +  ...
```

This is where Hall and Knight lost me. They say, from out of the blue, that each term after the second is equal to the sum of the two preceding terms multiplied respectively by the constants `_1` and `4`. Thus:

```
  7 = (_1*1) + (4*2)
```

or

```
  7 = _1 4 +/ . * 1 2
```

This implies that if we take any three consecutive terms *r, s, t,* they are related by:

```
   t = (_1 * r) + (4 * s)
```

which can be rewritten as:

```
   0 = (1 * r) + (_4 * s) + (1 * t)
```

In this equation the coefficients

```
   1 _4 1
```

of *r*, *s*, and *t* form the scale of relation of the infinite series. They are the coefficients of a quadratic polynomial written in ascending order:

```
   (1*y^0) + (_4*y^1) + (1*y^2)
```

Now this is all stated baldly in Hall & Knight, and I was thoroughly lost. How does one find the scale of relation, and what was the point of it? Luckily, the authors soon give the game away, noting that if a sufficient number of the terms of a series be given, the scale of relation may be found, and proceed to show how to do just that.

Suppose the first four terms of the series are, in order, *a*, *b*, *c*, and *d*. Assume then that the general term is arrived at by multiplying the two preceding terms by *p* and *q*, and adding. We are able to write the following pair of equations:

```
   c = (p*a) + (q*b)
   d = (p*b) + (q*c)
```

and then it is a simple matter to solve this linear system for *p* and *q* by writing

```
   'pq'=. (c,d) %. (a,b),:(b,c)
```

For example, if *a*, *b*, *c*, and *d* are 1 2 7 26 we write

```
   ]'pq'=. 7 26 %. 1 2,:2 7
_1 4
```

and we can form the scale of relation by appending a 1 to the negative of these:

```
   ]s=. 1 ,~ - 7 26 %. 1 2 ,: 2 7
1 _4 1
```

A pretty way to write this in J is to form a table t as follows:

```
   t =. 2 ]\ y =. 1 2 7 26
   t
1  2
2  7
7 26
```

and then we can write

```
   ({: %. }:)t
_1 4
```

So that we can form the scale of relation using the function `scr`:

```
   scr=. 1: ,~ [: - [: ({: %. }:) 2: ]\ ]
```

And use it to get our scale of relation:

```
   scr y
1 _4 1
```

What can we do with a scale of relation? Suppose we take the vector product of the scale of relation and any three successive terms of the infinite series, say 7 26 97

```
   a=. 1 _4 1 * 7 26 97
   a
7 _104 97
   +/ a
0
```

This sum will always be zero as a consequence of the way the infinite series and the scale of relation are interrelated. Consequently, if we do the polynomial multiplication of the scale of relation with the infinite series beginning with 1 2 7 26, we find that all the terms after the first two are zero:

```
   1     2     7    26    97   362 ...
   1    _4     1
   ---------------------------------- ...
   1     2     7    26    97   362 ...
        _4    _8   _28  _104  _388 ...
               1     2     7    26 ...
   ----------------------------------
   1    _2     0     0     0     0 ...
```

Since all the terms of this product after the first two are zero, and since we can ignore trailing zeros in a list of polynomial coefficients, we find that the infinite product of the scale of relation and the infinite series reduces to the linear polynomial:

```
   (1 * y ^ 0) + (_2 * y ^ 1)
```

In practice it is difficult to represent or work with infinite series, so we enable the process by using only the first two terms of the series. We can then do the polynomial multiplication of these two terms with the scale of relation, and take only the first two terms of the resulting product. Ordinary polynomial multiplication is given by:

```
   pm =. +//. @ (*/)
```

and our special infinite series multiplication by this scale of relation polynomial is given by:

```
   spm =. 2: {. pm
   1 _4 1 spm 1 2 7 26
1 _2
```

The next thought to convey to you is the most important one in the whole paper, so **PAY ATTENTION!**

Let me write the situation schematically:

$$\frac{\text{Infinite series} \times \text{Scale of Relation}}{\text{Scale of Relation}} \leftrightarrow \text{Infinite series}$$

That is, if I multiply and divide the infinite series by the scale of relation, I end up with the infinite series. But I know the numerator is simply a linear polynomial. So I can substitute the linear polynomial for the numerator and write:

$$\frac{\text{Linear Polynomial}}{\text{Scale of Relation}} \leftrightarrow \text{Infinite series}$$

This suggests that an infinite series of the kind we are describing can be represented as a rational polynomial whose numerator is the linear polynomial found as the product of the infinite series with its scale of relation, and its denominator is the scale of relation, and that this rational polynomial is fully equivalent to the infinite series. By this chicanery I have managed to encapsulate the whole infinite series in a rational polynomial. In J we represent a polynomial by a list of coefficients `c` bonded to the polynomial primitive `p.`, that is,

```
   c & p.
```

is a polynomial with coefficients `c`.

We can thus represent an infinite series by the rational polynomial function `gf`, using its product with the scale of relation as the numerator, and the scale of relation as the denominator.

```
   gf =. 1 _2&p. % 1 _4 1&p.
```

There isn't much we can do directly with `gf`, since the only meaningful arguments for it are those which make the infinite series converge, so we are restricted, if that is what we want to do, to arguments less than one in magnitude. But that isn't what we want to do. We are only interested in the coefficients of the terms in the series, and J provides us with the tool needed to find these, and that is the Taylor coefficient adverb (`t.`). Thus if we apply `t.` to `gf`, and apply this derived function to any non-negative integer argument, the result will be the corresponding coefficient:

```
   gf t. i. 12
1 2 7 26 97 362 1351 5042 18817 70226 262087 978122
```

Compared to the doubly recursive verb `A`, the time required by `gf  t.` is significantly less and its advantage in speed increases rapidly with the size of the argument.

I estimate `A  30` would have taken 15 hours to complete on my computer, versus the 1.2 seconds taken by `gf  t. 30`.

Partial Fractions

Hall & Knight discuss the relevance of partial fractions in handling recurrences, and work through some examples. This leads to the ability to derive an even simpler expression for the general term of the series. The method works as follows: separate the generating function into a sum of partial fractions with constant numerators and linear denominators. That is, find constants *a*, *b*, *A*, and *B* such that:

$$gf \leftrightarrow \frac{A}{1-ax}+\frac{B}{1-bx} \qquad (1)$$

The constants *a* and *b* are the roots of the scale of relation quadratic polynomial. These can be obtained using the polynomial rootfinder primitive, which is the monad of the verb `p.`, by

```
   ]'ab' =. , > }. p. 1 _4 1
3.73205 0.267949
```

You might recognize these roots as

```
   2 + %: 3 and 2 - %: 3
```

In (1) the denominators can be removed by multiplying each term by the scale of relation, giving:

```
   1 _2&p. <-> (A * (1 , -b)&p.) + (B * (1 , -a)&p.)
```

This linear system can be solved for *A* and *B* by writing

```
   ]'AB'=.1 _2 %. 1 1 ,:  -(b,a)
0.5 0.5
```

and now we can write a function `gt`:

```
   gt=. (A"_ * a"_^]) + (B"_ * b"_^])
   gt i. 10
1 2 7 26 97 362 1351 5042 18817 70226
```

The function `gt` is 5 times faster than `gf  t`.

But wait! Since *B* and *b* are each less than one, the right hand expression is always less than one and isn't really needed – we can replace it by a ceiling (`>.`). And since *A* is 0.5, we can replace it by halving (`-:`) giving us an even simpler expression:

```
   gtt =. >. @ -: @ (a & ^)
   gtt i.10
1 2 7 26 97 362 1351 5042 18817 70226
```

The function `gtt` is twice as fast as `gt`.

Index to Advertisers

Causeway Graphical Systems Ltd	10
Dyadic Systems Ltd	4
MicroAPL	126
Strand Software Inc	2
Vector Back Numbers	132

All queries regarding advertising in VECTOR should be made to Gill Smith, at 01439-788385, Compuserve: 100331,644.

Submitting Material to Vector

The Vector working group meets towards the end of the month in which Vector appears; we review material for issue n+1 and discuss themes for issues n+2 onwards. Please send the text of submitted articles (with diskette as appropriate) to the Editor:

Anthony Camacho,
11 Auburn Road, Redland,
BRISTOL, BS6 6LS
Tel: 0117-973 0036
Email: acamacho@cix.compulink.co.uk

Authors wishing to use Windows Write or Word for Windows should contact Vector Production for a copy of the Vector APL TrueType font and Vector APL typebox.

Camera-ready artwork (e.g. advertisements) and diskettes of 'standard' material (e.g. sustaining members' news) should be sent to Vector Production, Brook House, Gilling East, YORK YO6 4JJ.

Tel: 01439-788385
Compuserve: 100331,644.

British APL Association: Membership Form

Membership is open to anyone interested in APL. The membership year normally runs from 1st May to 30th April, but new members may join from 1st August, November or February if preferred. The British APL Association is a special interest group of the British Computer Society, Reg. Charity No. 292,786

Name:	______________________
Address:	______________________
	______________________
Postcode / Country:	______________________
Telephone Number:	______________________
Email Address:	______________________

Category (please tick box) to run from: 1st May ☐ August ☐ Nov ☐ Feb ☐

UK private membership	£12	☐
Overseas private membership	£14	☐
Airmail supplement (not needed for Europe)	£4	☐
UK Corporate membership	£100	☐
Corporate membership overseas	£135	☐
Sustaining membership	£430	☐
Non-voting UK member (student/OAP/unemployed only)	£6	☐

PAYMENT – in Sterling or by Visa/Mastercard/JCB only

Payment should be enclosed with membership applications in the form of a UK Sterling cheque to "The British APL Association", or you may quote your Mastercard, Visa or JCB number.

I authorise you to debit my Visa/Mastercard/JCB account

Number: ____ ____ ____ ____ Expiry date: __ | __

for the membership category indicated above,

☐ annually, at the prevailing rate, until further notice
☐ one year's subscription only

(please tick the required option above)

Data Protection Act: The information supplied may be stored on computer and processed in accordance with the registration of the British Computer Society.

Signature: ______________________ Send the completed form to:

British APL Association, c/o Rowena Small, 8 Cardigan Road, LONDON E3 5HU, UK

The British APL Association

The British APL Association is a Specialist Group of the British Computer Society. It is administered by a Committee of officers who are elected by a postal ballot of Association members prior to the Annual General Meeting. Working groups are also established in areas such as activity planning and journal production. Offers of assistance and involvement with any Association matters are welcomed and should be addressed in the first instance to the Secretary.

1995/96 Committee

Journal Working Group

Editor:	Anthony Camacho	0117-973 0036
Production:	Adrian & Gill Smith	Brook House, Gilling East, YORK (01439-788385)
Advertising:	Gill Smith	Brook House, Gilling East, YORK (01439-788385)
Support Team:	Jonathan Barman (01488-648575), Duncan Pearson (01653-618900), Richard and Adam Weber (01302-539761), Sylvia Camacho, Ray Cannon (01252-874697), John Searle (0181-858 6811), David Ziemann (0181-348 4039), Jon Sandles	

Typeset by APL-385 with MS Word 5.0 and GoScript

Printed in England by Short-Run Press Ltd, Exeter

VECTOR

VECTOR is the quarterly Journal of the British APL Association and is distributed to Association members in the UK and overseas. The British APL Association is a Specialist Group of the British Computer Society. APL stands for "A Programming Language" — an interactive computer language noted for its elegance, conciseness and fast development speed. It is supported on most mainframes, workstations and personal computers.

SUSTAINING MEMBERS

The Committee of the British APL Association wish to acknowledge the generous financial support of the following Association Sustaining Members. In many cases these organisations also provide manpower and administrative assistance to the Association at their own cost.

Causeway Graphical Systems Ltd
5 The Maltings, Castlegate,
MALTON, North Yorks YO17 0DP
Tel: 01653-696760
Fax: 01653-697719
Email: causeway@compuserve.com

Dyadic Systems Ltd
Riverside View, Basing Road,
Old Basing, BASINGSTOKE,
Hants, RG24 0AL
Tel:01256-811125
Fax:01256-811130
Email:sales@dyadic.com

Insight Systems ApS
Nordre Strandvej 119A
DK-3150 Hellebæk
Denmark
Tel: +45 42 10 70 22
Fax: +45 42 10 75 74
Email: insight@inet.uni-c.dk

MicroAPL Ltd
South Bank Technopark
90 London Road
LONDON SE1 6LN
Tel:0171-922 8866
Fax:0171-928 1006
Email:microapl@microapl.demon.co.uk

Dutch APL Association
Postbus 1341
3430BH Nieuwegein
Netherlands
Tel:03474-2337

Compass R&D Ltd
10 Frederick Sanger Road
Surrey Research Park
GUILDFORD, Surrey GU2 5YD
Tel:01483-302249
Fax:01483-302279

HMW Trading Systems Ltd
Hamilton House,
1 Temple Avenue,
LONDON EC4Y 0HA
Tel:0171-353 8900
Fax:0171-353 3325
Email:100020.2632@compuserve.com

Manugistics
2115 East Jefferson St
Rockville
MARYLAND 20852 USA
Tel: +1 (301) 984-5412
Fax: +1 (301) 984-5094
Email:aplsales@manu.com (US)
Email:intl@manu.com (International)

Soliton Associates Ltd
Groot Blankenberg 53
1082 AC Amsterdam
Netherlands
Tel:+31 20 646 4475
Fax:+31 20 644 1206
Email:sales@soliton.com