

VECTOR

**APL 93
Reports**



*The Journal of the
British APL Association*

A Specialist Group of the British Computer Society



ISSN 0955-1433

Vol.10 No.2 October 1993

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/PC, APL*PLUS II, IBM APL2/PC or Dyalog APL/W, or in documents from Windows Write (use the Vector TrueType font, available free from Vector Production).

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 1993-94

Category	Fee	Vectors	Passes
UK Private	£12	1	1
Overseas Private	£20	1	1
(Supplement for Airmail, not needed for Europe)	£8		
UK Corporate Membership	£100	10	5
Overseas Corporate	£155	10	
Sustaining	£430	50	5
Non-voting Member (Student, OAP, unemployed)	£6	1	1

The membership year 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 or Mastercard 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

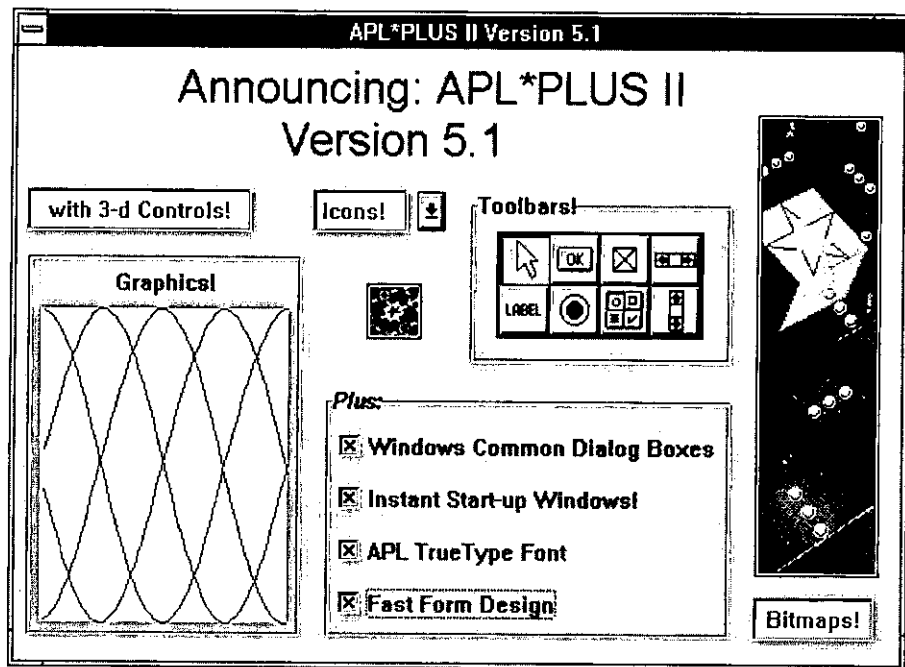
Advertisements in VECTOR should be submitted in typeset camera-ready format (A4 or A5) with a 20mm blank border after reduction. Illustrations should be photographs (b/w or colour prints) or line drawings. Rates are £250 per full page, £125 for half-page or less (there is a £75 surcharge per advertisement 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: 04393-385.

CONTENTS

		Page
Editorial: APL 93 at Toronto	Anthony Camacho	3
APL NEWS		
Quick Reference Diary		5
General Correspondence		
The APL 93 Russian Fund	Ben Best	7
News from Sustaining Members	Gill Smith	13
The Education Vector	Alan Mayer	17
REVIEWS SECTION		
APL Product Guide	Gill Smith	33
GDDME - a First Look	Adrian Smith	45
Helm — A Company-oriented Decision Support System (DSS)	Jon Sandles	49
RECENT MEETINGS: APL93 at Toronto		
Conference Roundup	Sylvia Camacho	56
APL93 Tutorial:		
Solving Wicked Problems with APL	Chris Lee	63
Programming the ISI Window Driver	Eric Iverson	67
Workshop: a GUI Standard for APL		87
APL in Satellite Surveillance	Jack Rudd	95
Postscript: "Don't Stop Thinking About Tomorrow"	Adrian Smith	110
GENERAL ARTICLES		
Mineswopper: GOTO Considered Futile	Adrian Smith	114
J Solution to New Scientist Enigma 685	David Ziemann	119
TECHNICAL SECTION		
Hackers' Corner: A Windows Task-Killer for APL	Duncan Pearson	126
At Play with J	Eugene E McDonnell	128
Migrating Mainframe APL2/TSO Applications to APL*PlusII/386	Allan Gay	130
A GDDM Simulation for APL*PlusII/386	Allan Gay	133
Index to Advertisers		143

We just couldn't leave well enough alone!



APL*PLUS[®] II for DOS and Windows[™] just got better! It's now even easier (and a lot faster!) to build applications using *all* of the features of Windows 3.1. There's even a Software Developer's Kit that shows you how to *easily* add third-party custom controls!

And don't forget that APL*PLUS II is *the only* APL system that provides you with a fully-integrated Debugger, Paradox Interface, full DDE access, Lotus

and dBase Import/Export, optimized Assembler functions, the User Command Processor, a Numeric Data Editor and the most powerful session manager available anywhere!

Get GUIing today! In the UK, phone Cocking & Drury at 071-436-9481. Elsewhere, call your local dealer or contact Manugistics, Inc. in the U.S. at (301) 984-5412, Fax (301) 984-5094 for the name of a dealer near you.

EDITORIAL: APL 93 at Toronto

by Anthony Camacho

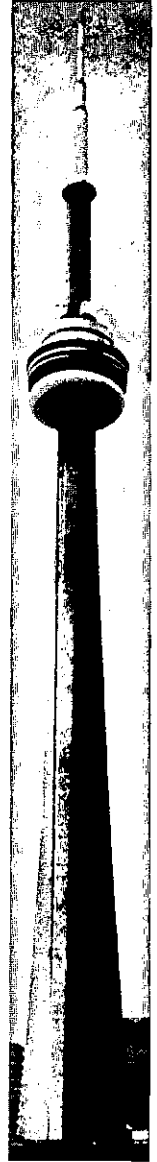
Toronto

May I thank the APL 93 Committee and their helpers on behalf of the APL community? APL 93 had the best programme of any conference I've attended. That made reporting it hard work. In four and a half days from 8:30 to 17:00 there were three sessions when I was not in one presentation or another; the reports seem to me more earnest than usual. Perhaps I tired myself out. I'm sure the people who ran it did.

Toronto is a very civilised place. Most of it is clean, it has excellent publicly run public transport, it tolerates minorities, there are few street beggars and they are not too threatening, it is safe to walk about at midnight, much of it is even beautiful and it has the most amazing warren of underground walkways and malls which extend about a mile north east from the CN Tower. You can see a lot of Toronto in a heat wave or the cold of winter without suffering from its climate.

The conference, too, was civilised. It ran with the minimum overt control. Many sessions even omitted a chairman. The weather was unusually hot and this made hurrying a bad idea; people often arrived late, especially for the more distant sessions. Luckily the presentation theatres were air conditioned but the room where Donald McIntyre's tutorial was held was not! It is hard enough without sweat tricking down one's face.

The civic reception was run-of-the-mill as these things go. The location was quite extraordinary. The city hall is a very avant garde building and I was glad of the chance to see inside it. The banquet at Casa Loma offered an excellent range of types of food. The place itself was fascinating to explore and the entertainments varied enough to please everyone. The picnic at Algonquin island was just the sort of relaxed occasion I needed. A pity so few people accepted the implied invitation on the bottom of the instructions how to get there "Yes, you may swim in the lake."



Vector

Nominally Jonathan Barman handed over the editor's chair to me after Volume 10 number 1 was printed. The material for volume 10 number 2 was nearly all collected and edited for me. Thank you Jonathan and the working group. Please do go on helping.

There is no charge for entries in the product guide. We are very keen to make it comprehensive and will welcome new entries. Even if you only provide your services in Vancouver or Valparaiso, please send us your entry for the product guide. Vector has a large international readership. What have you got to lose?

Little-known Feats of APLers:

Chris Lincoln, Joint Chairman of APL 89 in New York, has other talents. This July he sailed his windsurfer all round Fire Island in one day. Readers should know that Fire Island, though narrow, is over thirty miles long! Chris reckons that, with tacking, he sailed about eighty miles and the trip took nine hours.



Jim Brown Entertains at the Conference Picnic

Quick Reference Diary 1993-4

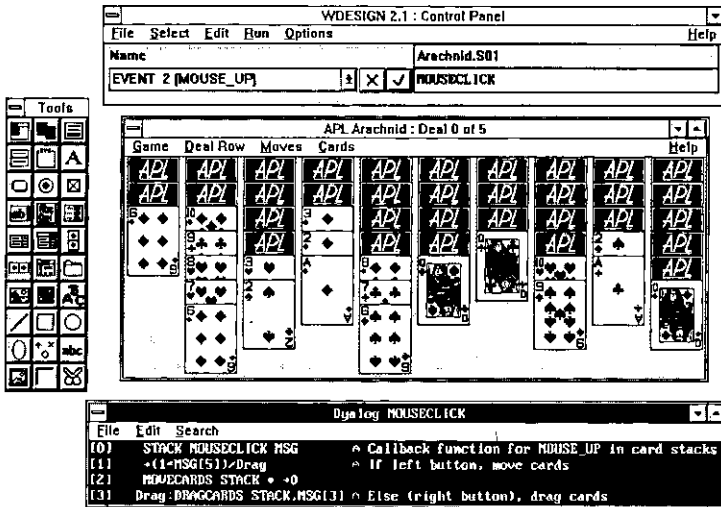
Date	Venue	Event
26 November	London (Venue tba)	BAA-GUI Workshop: All-day event (lunch is included). Space is limited - please contact Duncan Pearson at 143 Hull Road, YORK YO1 3JX (Tel: 0904-603510) to reserve a place. There will be a small charge (£75) to cover hire of equipment; target is one computer between 2 delegates.
	<i>Please use booking form on page 141 to reserve your place.</i>	
February 4 1994	IEE	BAA Meeting
March 6-8 1994	Phoenix, Arizona	Symposium on Applied Computing APL-Track Chair: Chris Lee (Manugistics) Papers and panel proposals by Sept 1st 93 Contact Chris Lee at (301) 984-5117
25 March 1994	IEE	BAA Meeting
27 May	IEE	BAA Meeting
18 - 22 July	EBMS/Swansea	APL in Business
11 - 15 September	Antwerp	APL 94
28 October	IEE	BAA Vendor Forum

British APL Association meetings are normally held in the IEE, Savoy Place.
Nearest tube outlets: Temple or Embankment.

Dates for Future Issues of VECTOR

	Vol.10 No.3	Vol.10 No.4	Vol.11 No.1
Copy date	3rd Dec 93	4th March 94	3rd June 94
Ad booking	10th Dec 93	11th March 94	10th June 94
Ad Copy	17th Dec 93	18th March 94	17th June 94
Distribution	January 94	April 94	July 94

The DEFINITIVE APL for WINDOWS™



- Produces snappy, professional-looking Windows applications.
- FREE Run-Time.
- Powerful, elegant and easy-to-use GUI support.
- Complete device-independent graphics, including bitmaps, icons.
- Built-in support for drag-drop.
- WDESIGN workspace for interactive GUI development.
- DDE support through Shared Variables.
- Direct access to Windows clipboard.
- On-the-fly calls to DLLs (eg Oracle, Microsoft SQL Server) using □NA.
- No workspace limitations.
- Supports all Windows printers with TrueType APL font.
- Workspaces portable to OSF/Motif, Windows NT and OS/2.

DYADIC

GENERAL CORRESPONDENCE

The APL 93 Russian Fund

From: Ben Best

20 Sept 93

I am happy to say that the APL 93 Russian Fund was a success. The Russian Fund paid transportation, Visa fees, conference registration, living expenses and arranged accommodation for seven Russians. Two additional Russians attending APL 93 received logistical support (and accommodation arrangement, in one case) from the Russian Fund, although they paid other expenses themselves. Andrei Kondrashev (Chairman of APL 92) has been living & working in Chicago and needed no help from the Russian Fund. Thus, there were a total of ten Russian delegates at APL 93.

The APL 93 Russian Fund total revenues were Cdn\$10,480.20 and total expenses were Cdn\$9,615.37. The APL 93 Russian Fund was closed-out by sending a cheque for US\$643.48 to the APL 94 Russian Fund.

Three key elements to the financial success of the Fund were:

- the fact that Russians could buy round-trip tickets to Montreal for under US\$500 (although these prices have been rising rapidly);
- the willingness of Toronto APLers to open their homes to Russian guests at no financial cost;
- the generosity of those who contributed to the Fund.

Special thanks go to the two largest contributors:

- The British APL Association, which contributed about US\$2,250.
- Security APL of Chicago, which contributed US\$1,000.

The nine delegates who came to APL 93 from Russia were Alexander Skomorokhov, Dmitri Lukyanov, Pavel Luksha, Oleg Luksha, Alexei Skurikhin, Nikolai Puntikov, Aibarsha Mukanova, Alexei Kononov and Andrei Pakhomov. A brief character sketch of each of these people is the best way to convey the true success of the APL 93 Russian Fund.

Alexander Skomorokhov (Sasha). Dr. Skomorokhov is the president of SovAPL. At APL 93 he presented the paper "Adaptive Learning Networks in APL2". "APL" in Russian is an abbreviation for "Atomic Underwater Submarine". Due to his background in nuclear energy, Dr. Skomorokhov casually bought a book with "APL" in the title at a second-hand bookstore in 1979. It was the textbook by Gilman & Rose. He began looking at the book at midnight one evening, and became so engrossed that he didn't stop reading until 8am next morning. APL was not implemented in Russian until 5 years later (by Andrei Kondrashev).

Dr. Skomorokhov used APL as a formal notation for algorithms, and still uses it for this purpose. He has written many types of software for nuclear power diagnostics in APL, including expert systems, time-series analysis, databases, statistics, etc. He learned of APL2 from Erkki Juvonen, and this discovery was almost as electrifying for him as his initial contact with APL. Teaching APL2 to children is now a favourite hobby of his.

Sasha is very gregarious and he counts many people in the APL community as personal friends. He relished the opportunity at APL 93 to meet these people and make new friends. He appreciated seeing the latest version of APL2 on OS/2. He especially liked the presentation "Undocumented Features of APL" by Tima Laurmaa.

Dmitri Lukyanov. Dmitri is Dr. Skomorokhov's colleague at the nuclear power plant in Obninsk, Russia (near Moscow). Like Sasha, he found Gilman & Rose to be an exciting book, and he enjoyed the problems even without the benefit of a computer. When APL became available on computer he wrote assembler programs to take data from diagnostic sensors and feed that data to APL software. With the advent of a market economy he has taken the initiative to use APL to write a menu system for a restaurant/casino part-time, in addition to his regular work.

At APL 93 Dmitri was particularly interested in learning more about the British ASL (APL Statistics Library). He was very impressed by the Windowing APL systems he saw at the exhibits. And he thought Robert Bernecky's tutorial on Parallel Computing was *great*. He hopes the Russian Fund will be unnecessary in a few years.

Pavel Luksha. Pavel is the 16-year-old son of long-time SovAPL member Oleg Luksha. At APL 93 he presented a paper on "Learning Modern Algebra" in which he demonstrated his attempts to solve the Problem of Irreducible Polynomials using APL. At the conference he particularly enjoyed the tutorials by Dr. McIntyre (Introduction to J) and Robert Bernecky (Parallel Computing) as well as the "Undocumented Features of APL" paper.

Oleg Luksha. Oleg is a lecturer at the Obninsk Study and Conference Centre for Nuclear Energy. He has been using APL since 1977 for teaching (to teach algorithms) and for scientific investigation (simulation of the process of dissolution of nuclear fuel). He has been interested in the ASL and was especially glad to meet Alan Sykes. But the high point of APL 93 for him was watching his son present a paper.

Alexei Skurikhin. Alexei presented the APL 93 paper "Identification of Parallelism in Neural Networks by Simulations in the Language J" with his University of Nebraska colleague Alvin Surkan. Alexei has no knowledge-of or interest-in APL. He only has experience in J. His main interest is neural networks, and he uses whatever tools he finds that can further his work. He uses J for prototyping small programs and uses C for large programs where speed of execution is more important. Alexei particularly liked Dr. McIntyre's "Introduction to J" tutorial, Robert Bernecky's "Parallel Computing" tutorial and Chris Lee's "Wicked Problems in APL" tutorial. He also liked the paper "Practical Statistical Computing using J" and he greatly enjoyed all the APL 93 social events.

Nikolai Puntikov. Nikolai was one of the key organizers of APL 92 and has continued to expand his expertise by organizing other, larger conferences in St. Petersburg (unrelated to APL). Nikolai took responsibility for co-ordinating the APL 93 Russian Fund in Russia (helping with visas, plane fares, money distribution and communications). Nikolai's expertise in linguistics was useful when Andrei Kondrashev needed to design a symbol table, keyboard and screen handler for the Russian implementation of APL. Nikolai still works at a language institute, but only uses APL for personal needs. He believes APL is a way of thinking, and he likes people who think in the APL way. He arrived at the final picnic after I left (I'm told he was on a book-buying spree) so I did not get his impressions of the conference.

Aibarsha Mukanova. Aibarsha was the only female in the group — and the only non-Russian. She lives and works in the former Soviet Republic of Kazakhstan, in Alma-Ata. She met Andrei Kondrashev in 1986 when she was doing her PhD thesis in Moscow. Currently she uses APL for differential equations used in modelling turbulent currents in the atmosphere. Studying the movement of particles over the Aral Sea is useful for forecasting dust and salt storms.

She said it was a great pleasure to learn of other scientists' work. She was very interested in Dr. Stephen Jaffe's plenary session on modelling petroleum chemistry. She thought Eric Iverson's tutorial "Windows GUI Programming in ISIAPL" was very good. Eric gave her an APL Windows system for her work,

which gives Iverson Software a toehold in Kazakhstan! Moreover, Ray Polivka has induced her to write up her work for Quote Quad.

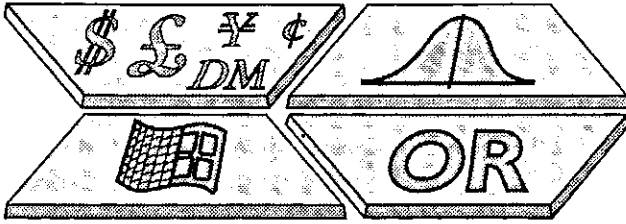
Alexei Kononov. Alexei received no money from the Russian Fund, but we did help him to find free accommodation — and we helped with logistics. I believe he was sponsored by the Ministry of Atomic Energy. He teaches algorithms by means of APL. He uses APL*PLUS. He won an APL*PLUS II system from Manugistics at APL 92, but he hasn't got the hardware to use it. At APL 93 he took a particular interest in "AGSS: A Graphical Statistical System", APLIWIN, "Wicked Problems in APL" and the (nearly) free run-time system of Manugistics. He thought David Eastwood's presentation on "Structured Functions with Operators" had some good ideas for producing more readable APL code for large systems. Alexei got Ken Iverson to autograph an APL 93 Conference Program.

Andrei Pakhomov. The APL 93 Russian Fund assisted Andrei with the logistics of registration, transportation and accommodation — but these were paid for in full by the company Andrei works for: EXIMA. Andrei had been a PASCAL programmer before meeting Andrei Kondrashev in 1986. Liking APL, he rewrote all his programs in the new language. With Boris Makeev, he developed an APL expert system for economic forecasting.

Then Andrei became more "practical" (he says), and started doing programming with the rapidly-growing new importing company EXIMA. Every aspect of EXIMA business operations is being programmed by Andrei in APL — tailoring the systems to the inflation, taxes and other idiosyncracies of the emerging Russian business environment. Andrei is held in almost god-like esteem by his company, which did about US\$25 million worth of business last year (that's a *lot* of Roubles!). At APL 93 Andrei became convinced that Dyalog APL is the best tool for creating a user-friendly environment for EXIMA's applications.

If the above character sketches are any indication, the APL 93 Russian Fund was not only a financial success, but a success in advancing APL in the rapidly-changing new world of Russia. I believe that contributions to the APL 93 Russian Fund were an investment which will reap many benefits for the APL community in the future.

Contributions to the APL 94 Russian Fund can be sent to: The APL 94 Russian Fund, c/o Madame L. Lemagnen, 174 Boulevard de Charonne, F-75020 Paris, France.



APL in Business

18th – 22nd July 1994
at University College, Swansea

Objective

To bring together the best practice in Finance, Statistics and Operational Research with the latest advances in APL and Windows.

Means

A 3-day Workshop mixing presentations with intensive hands-on learning with evening 'techie' free-for-alls.

Cost

The Workshop can cost you as little as £350 if you register early and stay on campus.

This could be the best value training in 1994!

PTO...

APL in Business

Provisional Programme

Tuesday 19th

Morning plenary session on OR and MIS
Afternoon Workshops (3 parallel sessions)
Evening Street Market and Technical Free-for-all

Wednesday 20th

Morning plenary session on Statistics (4 papers)
Afternoon Workshops (3 parallel sessions)
Another evening for the Techies!

Thursday 21st

Morning plenary session on Finance and EDI
Afternoon Workshops (3 parallel sessions)
Banquet (but the Labs are open til late)

Friday 22nd

Closing Plenary on the future of Distributed Applications

Facilities

The EBMS has 4 teaching labs, each equipped with 24 Windows-capable machines.

Any software developed for/during the workshops will be available for delegates to copy and take away.

Delegates may bring software for sale at the evening Street Market: the only restriction is on price – it must be under £100!

APL in Business is a joint venture by the European Business Management School and the British APL Association.

Contact: Mr J. Haydn Williams
Conference Administrator, EBMS,
University College, Singleton Park
SWANSEA SA2 8PP
Tel: 0792-295555 Fax: 0792-295626



The Windows Logo is a trademark of Microsoft Corporation

News from Sustaining Members

Compiled by Gill Smith

Kestrel Consulting

Our activities in America continue to expand with more consultants taking the opportunity to spend a year or two enjoying the sunnier climate while taking advantage of the American lifestyle. We still have openings for people who have strong APL2 skills. Vacancies outside APL in America include SAP, SMALLTALK and VISUAL BASIC.

At home the much talked about "green shoots of recovery" seem to be producing many new and interesting openings for those seeking permanent career moves as well as those looking for contract assignments. These opportunities exist in SAP and VISUAL BASIC as well as APL.

Manugistics Inc.

As many of you will have heard by now, Manugistics Group Inc. made an Initial Public Offering (IPO) on August 13th and is now trading on the NASDAQ as a public company. This achievement is a tribute to all our customers who have helped us grow. By the time you read this, our annual APL*PLUS Users' Conference will be over. Right now, however, we are deep into preparations with only a couple of days left before the welcome reception. This year's conference features presentations and workshops from developers and users with an emphasis on building applications. As soon as this is behind us, we'll be able to turn our attention to the Society of Actuaries conference in New York.

While the marketing folk are busy conferencing, the developers are hard at work. We've just shipped Version 5 of our APL*PLUS II for UNIX product (in four flavours) and now everyone is getting their heads down to work on a true Windows version of APL*PLUS II.

This will be a pure Windows version with no DOS remnants and will run under Windows 3.1 and Windows NT. We're working on a real Windows-hosted session manager and debugger and we'll be putting both the APL*PLUS interpreter and the APLGUI toolkit (re-written in C) in DLLs. This new Windows version is not intended to replace or make obsolete the DOS version of APL*PLUS II — there are still many, many people who are in no hurry to move to Windows. What it will mean is that users will be able to choose whether to

stay with DOS, move to Windows, or both. The good news is that the APLGUI applications you are writing now will be compatible with the new facilities — they'll just run a lot faster!

The best thing about APL*PLUS II for Windows is that all the work that is going into it serves a dual purpose. All the new features will also be used for our INCA project. This way, we will ensure that our APL*PLUS users get all the benefit of the new development as soon as possible. And this way, those of you who decide to move to INCA later on will find it easy to do so.

Dyadic Systems Ltd

Dyadic is pleased to report that, in terms of product interest, APL93 was its most successful conference yet. The company's stand was busy throughout the show and the attendance at Dyadic's Vendor Forum was greater than at any other in the 10 years since the product was launched. Perhaps though this had something to do with the refreshments that were provided!

The focus of the company's exhibit was the OSF/Motif implementation of Dyalog APL. This is functionally identical to Dyadic's increasingly popular Windows product and offers true portability of APL GUI applications across PC and UNIX-based systems. Dyadic was able to demonstrate several Dyalog APL/W workspaces, including Adrian Smith's PostScript graphics package running unchanged on an IBM RS/6000 under Motif. The Motif product should be available by the end of '93 and will coincide with a new release of Dyalog APL/W.

In addition to the OSF/Motif implementation, Dyadic announced new plans for Dyalog APL. Foremost among these is a commitment to provide compatibility with APL2. This will be implemented by having a new start-up option which will permit users to run Dyalog APL in "native mode" or "APL2-compatible" mode. Features such as "find", which present no conflict with existing Dyalog APL, will be available in both modes. APL2 compatibility will be introduced in stages, with the first of these included in the forthcoming Motif and Windows release at the end of the year.

Secondly, Dyadic announced binary compatibility between Dyalog APL workspaces and component files. This will permit Dyalog APL applications to be run in a heterogeneous network of PCs and UNIX systems and will greatly assist developers who wish to provide applications for both environments. Dyadic also promised to implement)COPY for GUI objects and intends at the same time to permit such objects to be stored on component files. Last, but not least, the

company announced plans to provide a fully CUA-compliant APL session and editor. These enhancements will all be included in the planned year-end releases. The company also indicated that it would develop a facility to allow third-party and user-written GUI objects to be accessed from Dyalog APL/W and announced long-term plans to incorporate more object-oriented features into Dyalog APL.

Soliton Associates Limited

Soliton Associates head-office relocated in July to new office space in the Toronto city centre:

Soliton Associates Limited
44 Victoria Street, Suite 2100
Toronto, Ontario
Canada M5C 1Y2 Tel: +1 416 364 9355; Fax: +1 416 364 6159

Soliton's mainframe development facility is now operated by a facility management company at a service centre in the Toronto area. Soliton's SHARP APL for MVS customer-base continues to enjoy worldwide network access to Soliton's Electronic Mail System as part of Soliton's 24-hour 365-day Technical Support Service.

Development of SHARP APL for MVS Version 21 is complete, and preparation for general distribution is in progress. The emphasis of Version 21 is on enhanced performance, operability and connectivity.

Version 1.5 of Soliton's MVSLink connectivity product is announced. MVSLink provides SHARP APL for MVS and SHARP APL for UNIX with access to data in the IBM MVS environment. It includes a high-capacity interface to IBM's DB2 relational database product.

A powerful new Intrinsic Function Facility is announced for SHARP APL for UNIX Version 4.03, allowing user-supplied interfaces to and from non-APL software. Standard intrinsic functions available with Version 4.03 include a TCP/IP Socket Interface.

The SHARE Europe '93 IBM User Conference will take place in October 1993 in The Hague, The Netherlands. The conference theme is "Client-Server, the Promise and the Reality", and on Tuesday 26 October there are presentations by Soliton Associates and other APL suppliers.

The APL93 International APL Conference in Toronto in August was a great success, and many new contacts were made. A strong Soliton is planned for APL94 in Antwerp, Belgium, in September 1994.

Contact Laurie Howard, Soliton Associates Limited, Amsterdam, The Netherlands, Tel +31 20 570 8733, Fax +31 20 570 8758, for information about Soliton products and services in Europe.

International Travel Solutions Ltd.

Freelancers
Moonlighters
Students and
Regular People

APL applications programming help wanted.
Especially STSC/Manugistics applications in the Travel or
Airline industry. C+Windows+OS/2 experience helpful.

Some maintenance, some sales, lots of opportunity
+ fun in a pure APL shop based in London with offices
also in USA, customers throughout EC and US.

Contact: C. Andrew Shepp at Claridge House, 29 Barnes High St, London SW13 9LW
Tel: 081 876 8666 Fax: 081 876 8660 USA Tel: +1 (314) 997 5498

THE EDUCATION VECTOR

October 1993

Editor Alan Mayer

This Education Vector has been reprinted from VECTOR Vol.10 No.2. VECTOR is the Quarterly Journal of the British APL Association. For more information about the British APL Association, please contact: Anthony Camacho, 11 Auburn Rd, Redland, BRISTOL, BS6 6LS Tel: 0272-730036.

Contents

Editorial	Alan Mayer	18
Review: Introduction to APL*PLUS PC	Alan Mayer	19
J-Ottings	Norman Thomson	21
An Improvement to the LOGGAMMA function	Norman Thomson	24
Cows and Bulls - A Solution	Ted Emms	25

Dr. Alan Mayer
European Business Management School
University College of Swansea
Singleton Park
Swansea SA2 8PP Wales, UK

Ysgol Rheolaeth Busnes Ewropeaidd
Parc Singleton,
Abertawe SA2 8PP

Tel: 0792-205678 Ext.4274 Fax: 0792-295626 Email: A.D.MAYER@SWANSEA.AC.UK

Editorial

Well, APL93 has come and gone. Many congratulations to the organisers in Toronto for a first class conference. A large contingent from the British APL Association was present, and several members gave papers. One of the spin-offs was an agreement between several of the local APL groups to exchange newsletters and where appropriate to reprint material. This type of cooperation can only be beneficial to the APL community throughout the world, and we look forward to fruitful collaboration in the future. If any other editors of APL newsletters would like to join us, please contact me and we will be pleased to welcome you aboard!

APL94 will be in Belgium in September 1994 — more of that at a later date, but if anyone would like a copy of the first announcement and “call for papers”, please drop me a line (or electronic equivalent).

Meanwhile two BAA events are worth noting. There is a GUI workshop in London on 26th November 1993, cost £75, details from the BAA Secretary, Duncan Pearson (0904-603510). In July 1994 there is an International Workshop “APL in Business” at Swansea University, jointly sponsored by the BAA and the European Business Management School, Swansea, for which most of our readers will have received an Announcement — I will gladly forward details to anyone who wants them. The continued development of GUI's, and the general standardization of interfaces between packages, make APL an invaluable ingredient in any serious business computing venture in the 1990's. These two events are designed to help business people make informed choices — and to realise what they are missing if they reject APL!

In this issue, as promised I have reviewed “Introduction to APL*PLUS/PC” by Maurice Dalois. My note on Cows and Bulls prompted Ted Emms to do a lot of work and produce a creditable effort at solving the 4 by 4 problem. I have included his (substantial!) solution in this edition, and note that he in turn has issued a challenge — can anyone improve on his coding of the tree structure which underlies his solution? Ted was restricted by his use of first-generation APL, but we will be interested in solutions using nested arrays too. Finally, Norman Thomson has written a short article on J, a language which some regard as the logical successor to APL, some regard as a valuable alternative, and others reject as too perplexing to be of any consequence! As usual, I await your views with interest.

If anyone would like to receive the rest of the journal Vector (144 pages!), from which Education Vector is taken, application forms for membership of the BAA are available from Rowena Small, 8 Cardigan Road, London, E3 5HU. UK subscription is currently £12 (students £6).

Please note the change in my email address (Edvec front cover) — the old one will work for some time, but the new one is preferable.

“Introduction to APL*PLUS/PC” by Maurice Dalois

reviewed by Alan Mayer

This book is an excellent introduction for APL beginners. It comes with a floppy disk containing an “educational version” of APL*PLUS/PC. This is similar to the Manugistics commercial product, with some limitations and omissions, and is quite adequate for the stated purpose of learning APL. At \$30 (plus shipping costs) for book and disk it is also ideal for many other educational purposes, such as teaching mathematics and statistics. I will start with a brief review of the software.

Installation of the system is very simple. The only startup decision the user has to make is the choice between three separate batch files for CGA, EGA and VGA. VGA and EGA work well. CGA is painfully slow, as the APL characters are produced in graphics mode. Only Epson compatible and HP LaserJet printers are supported by the Educational Version of APL*PLUS, and this choice can be made at startup by specifying a parameter for the batch file. Object sizes are limited to 64K and dimensions of arrays to 32767. A character vector cannot exceed 32767 characters; a numeric integer vector cannot exceed 32763 numbers; a numeric floating point vector cannot exceed 8190 numbers.

The system function `□MLOAD` (used to link up with non-APL programs) is not available. Terminal mode is not available (although a do-it-yourself transmit/receive program is described briefly in the text). Virtual workspace, extended memory usage by the interpreter, and file-sharing on networks are not available. Screen management is all there, as is file-handling. `□G` graphics are supported, but not the GSS interfaces. The user command processor works, but the documentation in the book is limited, so access to the full documentation is desirable.

Coming to the book itself, there are 190 pages of text, and a further 112 pages of appendices and an index. The text is divided into 9 chapters, which are easy to read, and take you rapidly but comprehensively through the main features of APL programming, from calculator mode to screen management, graphics and data files.

There are some typographical errors — words like beyond instead of beyond. None that I have found really matters: I mention them by way of a minor protest.

We all have spell-checkers these days. I know they can be annoying, but used sensibly they should help eradicate these irritations. (Yes, I know this is dangerous stuff — send me all the typos you find in Education Vector and I will humbly apologise for every one!)

From chapter 3 onwards, various utility functions are introduced in the text. They are listed as they are introduced. They also appear in Appendix C, and are available on the disk. You would be forgiven for assuming that functions with the same name, taken from any of these sources, would be identical — but you would be wrong. In at least one case (a function called ASKN) all three listings are different! Most of the functions perform correctly, so the inconsistencies are again little more than an irritation.

In the case of a function called DISPLAY, however, there are some problems. The disk version corresponds to the appendix, but not to the text. The problem is that only the text version gives the results claimed in the text. Working through Chapter 4, on Screen Management, we are advised to copy DISPLAY from the workspace UTIL and perform a few simple tasks. The first task involves displaying a “blinking” statement on line 24 of the screen. Unfortunately line 24 is a Status line, so the statement cannot be seen unless you scroll down to it. Then you display other statements at various positions on the screen, and that is when the disk (and appendix) versions let you down.

It is sad that these flaws mar what is otherwise an excellent text. Once you realise that when things don't work it may not be your fault, you can benefit greatly from an exploration of Screen Management in Chapter 4, Graphics in Chapter 5 and Data Files in Chapter 6. Chapter 7, entitled Miscellaneous Techniques, introduces Recursion and the User Command Processor. A very brief part of Chapter 7 also discusses communications with printers, other computers, plotters and digitizers — sufficient information to whet the appetite and perhaps to set the determined explorer off on the right track. The last two chapters present miscellaneous problems (to which solutions are provided in an appendix, together with solutions to problems posed throughout the text) and a case study on processing election results.

Appendix A is particularly valuable as a concise reference manual. Entitled “Definitions” it includes a complete list of all primitive functions and operators, together with examples of their various uses. This is followed by a complete list of all system functions, variables and commands.

Book and disk are available from EducAPL Inc, 1120 av du Parc, QUEBEC, Que, CANADA, G1S 2W7. They are undoubtedly an important addition to the APL educational library.

J-ottings

by Norman Thomson

Anyone attending APL93 and paying any attention at all to the presence of J would be struck by the divide between the experts and the much larger number of people to whom J, even after a reasonably long acquaintance, remains a source of intriguing perplexity. There can be no question that mastery does not come in the same blinding fashion that it did with APL, and yet the challenge does not go away as realisation dawns that the road is indeed harder — after all there is always the interpreter to prove that the more mysterious outcomes are backed by an incontrovertible logic.

Of course the sheer richness of meaning with which the mundane set of ASCII symbols have been imbued is both a source of marvel on the one hand and despair on the other as the task of memorising even a subset of the language is contemplated. As if this were not enough there are profound differences which divide the philosophy of J from that of APL and indeed of any other extant programming language, and it may be that the proponents of J have understated some of the more fundamental areas of understanding which are pre-requisite to a good grounding in J.

The object of this occasional column is to offer some "ground level" thoughts which may help to make J more accessible to readers of this magazine who might have been tempted to dismiss J as either an untidy transliteration of APL on the one hand, or an impossibly abbreviated symbolism on the other.

To start with, APL functions have a property of replaceability, by which is meant that if it is observed that

```

      +÷5
0.2

```

the +÷ can be replaced with a function

```

[0] Z+FN R
[1] Z+÷R

```

to obtain

```

      FN 5
0.2

```

A defined verb in J does not however possess the replaceability property. + % 5 is indeed 0.2 but define fn =. + % and what happens is:

```
fn 5
5.2
```

The verb combination (+ %) is quite different from the two verbs in succession. A rough parallel in English might be the sentence "Double cross the road" which could be understood as an instruction to cross the road twice, quite different in meaning from the admittedly less likely utterance "Doublecross the road."

What has happened in defining fn is that + % has become the verb combination (+ %) and it is easily verifiable that (+ %)5 is indeed 5.2. This form of verb combination is what is known as a hook. In the absence of a left argument the right argument doubles up as left argument. The hook is essentially a lop-sided affair in the sense that the rightmost verb is applied to the right argument only, following which the left verb is applied to both. It is imperative that two arguments must be present. Without a hook construction (e.g. as in APL) "I double cross the road" describes my declamation "Cross the road, cross the road!" With a hook "I double cross the road" would mean that I cross the road, and then do so a second time, a sort of no-op! Like all analogies this one should not be pursued too far, however the principle should be clear, namely that replaceability as in APL function definition leads to a simpler but less rich form of language.

Function replaceability can if required be realized in J as sequential function composition which is described by the conjunction @.

We thus have

```
gn =. + @ %
gn 5
0.2
```

The difficulty of the mental hurdle over which hook requires us to jump may be reflected by the fact that it is hard to find compound words in English which are made up of two verbs. Others which come to mind are *blow-dry* and *cross-check*. Under sequential composition, to blow dry one's (wife's) hair would presumably mean making an unwanted disturbance to an already complete coiffeur; add the hyphen and things change a lot! Again cross check (no hyphen) suggests placing a final cross following an audit, which is by no means the same as cross-check! Of course it can be argued that the components in the foregoing word are not being used in a purely verbal way, which merely underlines the difficulty of finding verb compositions in English, and thus of relating hook to normal

linguistic experience. I suggest it is good practice to separate uncomposed J verbs with spaces, and to remove these only when the verbs are combined either in verb definition, or with parentheses in immediate execution. This is entirely consistent with separating distinct words in English with spaces, whereas providing a hyphen gives, at least in some instances, a rough equivalent of hook.

If *hook* is connected, albeit tenuously, with our experience of English words, *fork* is connected instead to our most basic parsing experiences in English grammar. A fork is essentially a tree structure, something quite foreign to APL, but common enough in the "subject verb object" structure of simple sentences. Take the "world's favourite one-line" as an example of a fork:

```

mean =. +/%#
w =. 1 4 2 2
mean w
2.25          or equivalently
(+/%#)w
0.25

```

A fork means take the outside verbs, *+* and *#*, apply them separately, and then — grand climax! — apply the one in the middle.

Now consider something more complex. First in APL define

```

[0] Z+L F2 R
[1] Z+1+R-L

```

a function which should not be too difficult for readers of this magazine to comprehend.

J is unlike APL in that the same verb *>*: doubles up as "greater than or equal to" in the dyadic case, and "1+," that is "increment" in the monadic case. So the sentence

```
4 >: ] - [ 5
```

means (executing the verbs from right to left)

```

take the left argument, namely 4
negate it to give _4
take the right argument of this, which is still _4
is 4 greater than or equal to _4, answer 1

```

The sentence above is thus a boolean expression. Now consider two verbs:

```

f1 =.>: ] - [
f2 =.>: @ ] - [

```

In the case of `f1` the verbs combine from the right, three verbs in a row combining to form a fork for which the outside verbs are "take left argument" (`⌈`) and "take right argument" (`⌊`). These are applied separately, and then the minus applied to their results. Call this verb combination "range". What is left is a two-verb combination which as we saw above must be a hook. So the question answered by `f1` is:

is the left argument greater than or equal to the range?

Contrast this with `f2` where there is sequential composition and the monadic meaning of `>`: has priority over the dyadic. `f2` thus means:

increment the range in line with the APL function of the same name.

We have already progressed to J verbs whose definitions involve five ASCII characters — a large leap indeed for one session!

An Improvement to LOGGAMMA

(Step by Step Analysis of Variance, Education Vector, July 1993)

From: Norman Thomson, Greenock, Scotland.

In the first place the name of the function is misleading since what *LOGGAMMA* delivers is log of gamma of half the argument. An excellent approximation for true log gamma can be obtained from Feller's extension to Stirling's formula for log factorial, namely

$$\ln n! = .5(\ln 2n) + (n+.5)(\ln n) - n + \frac{1}{12n} - \frac{1}{360n^3}$$

(see *An Introduction to Probability*, William Feller). Here it is in APL:

```
[0] Z←LOGFACT N
[1] →0 IF 0=Z+N
[2] Z+((.5+0,N),÷¯1,12,¯360)+.×(⊙2),N),N,÷¯N,N*3
```

A final trivial adjustment must be made to allow for the fact that $\Gamma(n) = (n-1)!$

For the purposes of *FTAIL* the above routine is an unnecessary refinement. However there are circumstances where it is invaluable, for example in calculating binomial probabilities with large parameter values, or in applying the Fisher Exact Test.

Cows and Bulls — A Solution

From Ted Emms, Kenley, Surrey.

I read the article "Cows and Bulls" (Education Vector, July, 1993) and got hung-up on the challenge in the last paragraph. I concentrated on the 4 *COWBULL* 4 case, which turned out to be a wise move since my first efforts kept informing me that I was out of memory. First let me give you an example of a *RUN*, where the correct solution is *CABC*:

INTRO

BULLS & COWS

=====

From the letters ABCD you choose a "word" composed of 4 letters, e.g. BCAD or BBCB. The computer attempts to guess the "word". To each guess given by the computer you respond with the number of bulls (correct letters in correct places) and the number of cows (correct letters in incorrect places).

You input the two numbers together. Thus if the number of bulls is 2 and the number of cows is 0, you input 20 and press <RET>.

The computer arrives at your "word" in as few attempts as possible.

Make your choice ready to play...

Guess No.1 is AABC Bulls and Cows (two nos. to be inputted together)? 30

Guess No.2 is ADBC Bulls and Cows (two nos. to be inputted together)? 21

Guess No.3 is AABD Bulls and Cows (two nos. to be inputted together)? 20

Guess No.4 is BABC Bulls and Cows (two nos. to be inputted together)? 30

Guess No.5 is CABC Bulls and Cows (two nos. to be inputted together)? 40

----- It took 5 moves to find CABC -----

The problem is solved in five gos. Most problems are solved in four and a few in less than that. The program is run by calling *INTRO*. This calls the routines *INIT*, *INIT1*, *INIT2*, *INIT3*, *INIT4*, *INPUT* and *PLAY*. I will explain those later, but first let me say how I approached the problem.

With the four letters *ABCD* to be used in four positions this gives 256 possible goals or targets. I decided to work with numbers rather than letters so I used the vectors 0 0 0 0 and 0 0 0 1 ... up to 3 3 3 3. The possible replies (*BULL,COW*) to testing a *T(RY)* against a *G(OAL)* can be 00, 01 etc. up to success which is 40. I decided to work with a *V(ALUE)* defined by $V+5 \times BULL+COW$. The vector $C+0$ 1 2 3 4 5 6 7 8 10 11 12 15 20 gives all the possible values of *V*. The routine *G TEST T* (see Appendix 2) is straightforward and corresponds to the approach in the original article with one slight difference. To get the *COW* score I have used the membership function.

There are 256 possible goals. Suppose we have (by applying tries) reduced the number of possible goals to be investigated to just a few. We will take a simple example putting $K=23$ 42 71 197 199 206. K is a vector holding those possible numbers. If we try the numbers K against the first of these by repeated application of 23 TEST T we will get a series of results (V s). In this particular case we would get 20 5 12 8 11 3. Indeed if we repeat this for the other numbers we get the table:

$G \setminus T$	23	42	71	197	199	206
23	20	5	12	8	11	3
42	5	20	1	1	1	6
71	12	1	20	12	15	7
197	8	1	12	20	15	10
199	11	1	15	15	20	11
206	2	6	6	10	11	20

Note that the table is not symmetrical about the diagonal which arises from the fact that X TEST Y is not necessarily equal to Y TEST X .

From the table we see that if $T=23$ each G gives rise to a different V . In the case of $T=42$ there are just four different V s with $V=1$ being repeated 3 times. With $T=199$ we get four different V s with both $V=11$ and $V=15$ being repeated twice.

We need a criterion for the best choice of T . We want the T which gives rise to the greatest number of different V s. (This is akin to the divide-and-conquer technique used in search procedures.) If there is a tie then we choose between the contending T s by choosing the T having the minimum number of repeats. I have, arbitrarily selected a parameter given by $((\text{No. of different } V\text{s}) * 2) \div (\text{No. of repeats})$ to make that selection. I wrote a routine *BEST* (not listed) to work this all out, which (eventually) gives the result that the best T is 6 (this corresponds to *AABC* in letters).

Having got the best T there remains the problem of finding out what possible numbers remain for each of the possible V s. For this I wrote another program *RG* (also not listed), which works with K having first been specified. Thus if we wish to know the G s still possible after the first guess ($T=6$) we specify K as $K \leftarrow 1 + i 256$ and call *RG*. When it asks for the try you input 6 and the program gives the answers, albeit slowly. Using routines *BEST* and *RG* in an orderly fashion we can build up a table giving a tree-like structure of what T should be used given a V . The results are listed in Appendix 1.

Thus if your nominated "word" is 235 then when the first try is 6 you should get $V=1$. Looking at $V=1$ in level 1 we see that the next try should be $T=253$. This

will give a response 7, so from level 2 we see we should put $T=187$. This gives $V=12$ and the table tells us to use $T=175$ in the next guess. The table gives $V=12$ and we respond with $T=235$ which is recognised ($V=20$) as the original "word".

Having done all the hard work there now remained the task of writing the proper program using the information in the table. But how does one enter a tree in APL? I confess I thought about this for some time and finally came up with a solution. Whether it is the "correct" way I don't know.

The 13 T values in level 1 (255 253 159...) as a vector X . For $V=1$ in level 1, I defined the 13 T values in level 2, (0 0 0 171...) and defined a vector $X17$ (i.e. X and the V of level 1). Similarly $X17$ (1 for level 1, 7 for level 2) I defined as the 175 in the $V=12$ position. i.e. $X17+(11\rho 0)$, 175. Finally in column 4 I defined $X17C$ as 235 in the $V=12$ position (the C in $X17C$ means 12 using hex notation). Using this method I created all the vectors contained in $INIT1$, $INIT2$, $INIT3$ and $INIT4$. I had to use 4 routines to contain all the information to overcome the limitation of only 30 program lines. Finally I wrote the proper program, *PLAY!*

If there is a better way to handle trees I should appreciate hearing about it.

Appendix 1

Level 1	Level 2	Level 3	Level 4	Level 1	Level 2	Level 3	Level 4
0 255				5 190	0 85		
1 253	3 171			2 87	12 213		
	7 187	12 175	12 235	5 117	15 215		
	8 95			6 119	6 207		
	10 93				12 245		
	11 251	12 191	12 239		15 247		
	12 127	8 223		8 234			
	15 125	12 221		10 63	0 170		
				12 238	12 250		
				15 174	10 254		
					12 186		
2 159	2 105	240		6 122	1 192	15 195	12 204
	4 121	11 233		2 29	11 205		
		15 249			15 31		
	6 89	0 243	252	3 47	1 149		
	7 107	10 169			5 165		
	8 123	11 109			12 143		
		4 237		4 151	8 229		
		8 217			11 231		
		11 185		5 51	15 167		
	10 153				12 60		
	11 91	4 173		6 61	15 48		
	12 189	4 219			3 203		
		8 111			8 79		
	15 155	11 157			11 77		
				7 43	3 222		
					5 101		
3 115	1 168			8 103	12 139		
	3 172	12 232			4 158		
		15 236			8 181		
	4 220						
	5 160						
	6 184						
	7 188	8 224					

		12 248	
8	92	8 209	
		11 208	
10	80	5 163	
		10 176	
		15 81	
11	227		
12	124	4 211	
		8 241	
15	83	10 179	
		11 112	
		12 113	

4	147	3 104	
		4 108	12 120
		7 88	8 97
			11 96
		8 216	8 225
		11 152	8 161
		12 99	4 156
			8 177
		15 144	15 145

Level 1	Level 2	Level 3	Level 4
	8 28	7 57	
		8 49	
	10 56	6 64	
	11 76		
	12 212		
	15 84	10 244	

8	114	3 24	7 164
			11 148
		4 228	
		7 33	4 72
			8 100
			12 129
		8 180	
		11 146	
		12 210	
		15 82	15 98

10	182	0 0	
		1 3	
			12 12
			15 15
		5 21	12 69
		6 23	2 206
			5 42
			8 197
			11 199
			12 71
		7 46	8 202
		10 53	1 138
			6 86
			10 62
			15 55
		11 58	6 102
			8 142
		12 230	
		15 118	10 166
			11 150
			15 246

			11 183
		10 59	
		11 94	10 154
		12 110	8 218
		15 90	10 126
			15 106

7	116	1 162	
		2 35	11 226
			12 131
		3 17	7 137
			10 41
			15 25
		4 19	4 141
			7 45
			8 193
			15 27
		5 32	8 136
			12 128
			15 40
		6 44	4 178
			8 200
			12 140
		7 65	8 16
			10 75
			15 67
			10 73

Level 1	Level 2	Level 3	Level 4
11 50	2 68		
	3 133	10 196	
	4 135		
	6 20	11 37	
	7 13	6 74	
		7 39	
		10 8	
	8 11	7 78	
	10 1	6 26	
	11 30	6 130	
		7 52	
	12 194		
	15 34		

12 66	4 36		
	8 9	8 132	
	12 18		

15 54	10 2	10 5	
		11 4	
		15 10	
	11 7	10 70	15 134
		11 14	
	12 198		
	15 22	15 38	

Appendix 2 — Program Listings

```

[0]  INTRO
[1]  '                BULLS & COWS'
[2]  '                ====='
[3]  ' From the letters ABCD you choose a "word" composed of 4 letters,'
[4]  ' e.g. BCAD or BBCB. The computer attempts to guess the "word".'
[5]  ' To each guess given by the computer you respond with the number'
[6]  ' of bulls (correct letters in correct places) and the number of cows'
[7]  ' (correct letters in incorrect places).'
[8]  ''
[9]  ' You input the two numbers together. Thus if the number of bulls is'
[10] ' 2 and the number of cows is 0, you input 20 and press <RET>.'
[11] ''
[12] ' The computer arrives at your "word" in as few attempts as possible.'
[13] ''
[14] ' Make your choice ready to play...'
[15] 2 10 ' '
[16] ' Press <RET> when ready....'
[17] 0
[18] 5 10 ' '
[19] INIT
[20] PLAY

```

```

[0]  PLAY
[1]  CNT+1
[2]  T+6
[3]  D+'X'
[4]  LPPLAY:' '
[5]  'Guess No.',(V CNT),' is ',E[1+FTT]
[6]  IN: INPUT 'Bulls and Cows (two nos. to be inputted together)? '
[7]  +(2*pI)/ERR
[8]  V+(1I[2])+5*1I[1]
[9]  +(V=20)/SUCC
[10] M+1D
[11] T+M[C,V]
[12] D+D,A[C,V]
[13] CNT+CNT+1
[14] →LPPLAY
[15] SUCC:
[16] 280 '- '
[17] 'It took ',(V CNT),' moves to find ',(V E[1+FTT])
[18] 280 '- '
[19] +0
[20] ERR: 'Incorrect input - try again!'
[21] →IN

```

```

[0]  INIT
[1]  D+'X'
[2]  T+6
[3]  F+4p4
[4]  C+0 1 2 3 4 5 6 7 8 10 11 12 15
[5]  A+'012345678ABCF'
[6]  E+'ABCD'
[7]  M+255 253 159 115 147 190 122 116 114 182 50 66 54
[8]  INIT1
[9]  INIT2
[10] INIT3
[11] INIT4

```

```

[0]  INIT1
[1]  X+255 253 159 115 147 190 122 116 114 182 50 66 54
[2]  X0+255 0
[3]  X1+0 0 0 171 0 0 0 187 95 93 251 127 125
[4]  X17+(11p0),175
[5]  X17C+(11p0),235
[6]  X1B+(11p0),191
[7]  X1BC+(11p0),239
[8]  X1C+(11p0),223
[9]  X1F+(11p0),221
[10] X2+0 0 105 0 121 0 89 107 123 153 91 189 155
[11] X22+240 0
[12] X24+(10p0),233 0 249
[13] X26+243 0 0 0 0 0 0 0 0 169
[14] X27+(10p0),109
[15] X28+0 0 0 0 237 0 0 0 217 0 185
[16] X2B+(4p0),173
[17] X2C+(4p0),219 0 0 0 111
[18] X2F+(10p0),157
[19] X260+(11p0),252
[20] X3+0 168 0 172 220 160 184 188 92 80 227 124 83
[21] X33+(11p0),232 236
[22] X37+(8p0),224 0 0 248
[23] X38+(8p0),209 0 208
[24] X39+(11p0),209 0 208
[25] X3A+(5p0),163 0 0 0 176 0 0 81
[26] X3C+(4p0),211 0 0 0 241
[27] X3F+(9p0),179 112 113
    
```

```

[0]  INIT2
[1]  X4+0 0 0 104 108 0 0 88 216 0 152 99 144
[2]  X44+(11p0),120
[3]  X47+(8p0),97 0 96
[4]  X48+(8p0),225
[5]  X4B+(8p0),161
[6]  X4C+(4p0),156 0 0 0 177
[7]  X4F+(12p0),145
[8]  X5+85 0 87 0 0 117 119 0 234 63 0 238 174
[9]  X52+(11p0),213 215
[10] X56+(6p0),207 0 0 0 0 245 247
[11] X5A+170 0
[12] X5C+(11p0),250
[13] X5F+(9p0),254 0 186
[14] X6+0 192 29 47 151 51 61 43 103 59 94 110 90
[15] X61+(12p0),195
[16] X61F+(11p0),204
[17] X62+(10p0),205 0 31
[18] X63+0 149 0 0 0 165 0 0 0 0 0 143
[19] X64+(8p0),229 0 231 0 167
[20] X65+(11p0),60 48
[21] X66+0 0 0 203 0 0 0 0 79 0 77
[22] X67+0 0 0 222 0 101 0 0 0 0 0 139
[23] X68+0 0 0 0 158 0 0 0 181 0 183
[24] X6B+(9p0),154
[25] X6C+(8p0),218
[26] X6F+(9p0),126 0 0 106
    
```



```

[0]  INIT3
[1]  X7+0 162 35 17 19 32 44 65 28 56 76 212 84
[2]  X72+(10p0),226 131
[3]  X73+(7p0),137 8 41 0 0 25
[4]  X74+0 0 0 0 141 0 0 45 193 0 0 0 27
[5]  X744+(11p0),201
[6]  X75+(8p0),136 0 0 128 40
[7]  X76+0 0 0 0 178 0 0 0 200 0 0 140
[8]  X764+(12p0),242
[9]  X77+(8p0),16 75 0 0 67
[10] X77F+(9p0),73
[11] X78+(7p0),57 49
[12] X7A+(6p0),64
[13] X7F+(9p0),244
[14] X8+0 0 0 24 228 0 0 33 180 0 146 210 82
[15] X83+(7p0),164 0 0 148
[16] X87+0 0 0 0 72 0 0 0 100 0 0 129
[17] X8F+(12p0),98
[18] XA+0 3 0 0 0 21 23 46 0 53 58 230 118
[19] XA1+(11p0),12 15
[20] XA5+(11p0),69
[21] XA6+0 0 206 0 0 42 0 0 197 0 199 71
[22] XA7+(8p0),202
[23] XAA+0 138 0 0 0 0 86 0 0 62 0 0 55
[24] XAA6+(12p0),150
[25] XAB+(6p0),102 0 142
[26] XAB6+(10p0),214
[27] XAC+(11p0),230
[28] XAF+(9p0),166 150 0 246

```

```

[0]  INIT4
[1]  XB+0 0 68 133 135 0 20 13 11 1 30 194 34
[2]  XB3+(9p0),196
[3]  XB6+(10p0),37
[4]  XB7+(6p0),74 39 0 8
[5]  XB8+(7p0),78
[6]  XBA+(6p0),26
[7]  XBB+(6p0),130 52
[8]  XC+0 0 0 0 36 0 0 0 9 0 0 18
[9]  XC8+(8p0),132
[10] XF+(9p0),2 7 198 22
[11] XFA+(9p0),5 4 0 10
[12] XFB+(9p0),70 14
[13] XFBA+(12p0),134
[14] XFF+(12p0),38

```

```

[0]  INPUT R
[1]  I+R
[2]  I+(pR)+,I

```

```

[0]  G TEST T
[1]  AA+FTG
[2]  BB+FTT
[3]  BULLS+~/I+AA=BB
[4]  AA+{~I}/AA
[5]  BB+{~I}/BB
[6]  COWS+~/BBεAA
[7]  V+COWS+5×BULLS

```



RENAISSANCE DATA SYSTEMS
Enlightenment Thru Information Processing

Specializing in Books and Software on APL, J
and other Curiosities of merit.

HAVE YOU GIVEN A COPY OF IAPL OR J TO A TEACHER OR STUDENT?

Name: _____ Date: _____
Street: _____
City: _____ State: _____ Postal Code _____
Country: _____ Telephone _____

For a copy of our most recent catalog,
please send a self-addressed, legal-
sized envelope (stamped if from U.S.)
or mail this form to:

Renaissance Data Systems
Park West Finance Station
P. O. Box 20023
New York, New York 10025-1510

Learn APL

APL Shareware/Demos and Related Publications

APL and Mathematics

APL AS A TOOL OF THOUGHT Proceedings

Other Important Sources of APL Information.

APL History

APL References and Techniques

Special Subjects in APL

APL Interpreters and Software

J=: A Powerful Dialect of APL

Other Software for Use with APL

To Russia with Love and for APL92

Other Curiosities of Merit

**I-APL: An international voluntary project of individuals and
companies seeking to share their love of APL.**

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.

We do depend on the alacrity of suppliers to keep us informed about their products so that we can update the Guide for each issue of VECTOR. Any suppliers who are 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

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: 04393-385

COMPLETE APL SYSTEMS

COMPANY	PRODUCT	PRICES(£)	DETAILS
Active Workspace	AWL486	1,950	486 based 33MHz PC, 140MB Disk, 4MB RAM, VGA Colour. (inc. 1 year on site maintenance.)
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 18" 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 18" 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, GDMM and the full line of Interprocess APL2 enhancements.
MicroAPL	IBM RS/6000	12,000+	POWER range of RISC systems running AIX. Dumb terminal or graphical interface.
	Aurora	20,000+	Multi-user APL computer using 68020 CPU. Std. configuration 2Mb RAM, 16 RS232 ports, 68 Mb hard disc, 720K diskette
Optima	IBM Compatible	poa	Complete PC-based station, APL interpreters & all support eq't

APL INTERPRETERS

COMPANY	PRODUCT	PRICES(£)	DETAILS
Active Workspace	DYALOG APL DOS 386	poa	Dyadic's PC 386 APL Interpreter.
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 Dyalog APL	poa 1000-10,000	2nd generation APL for Unix systems
APL2/PC		poa	IBM's APL 2 for the PC.
		poa	K is an APL-like language
Atlantis Software	Analytic Platform (K)	poa	
Cocking/Drury	APL*PLUS PC Rel 10	410	STSC's full featured APL for IBM's and compatibles - Version 10 includes the Quad-NA facility to interface to non-APL software, support for Microsoft Windows and mouse devices. The User-command processor has been built in to the interpreter.

	APL*PLUS PC Run-Time	175 for 5	Closed version of the interpreter for developers, prevents user exposure to APL.
	APL*PLUS PC Developer System	950	Gives rights to distribute an unlimited number of copies of Run-Time application.
	APL*PLUS II System	1200	High powered APL Interpreter for the 80386 chip. Price includes one year's maintenance and free upgrades - volume discounts
	APL*PLUS II Developer System	3200	Unlimited distribution of APL*PLUS II Run-Time applications!
	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 Aitos, 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 - 11	ISO conforming interpreter. Supplied only with manual (see 'Other Products' for accompanying books).
	I-APL/BBC Master	8	As above
	I-APL/Archimedes	11	As above
	I-APL/Macintosh	13	As above
	Iverson Software Inc		I-APL is the UK agent for all ISI products, including APLWIN and JWIN for PC and many other machines.
I-APL/ISI	APLWIN/386	50	Windows APL Including manual
I-APL/ISI	JWIN/386	16	Including <i>Dictionary of J</i> and <i>Introduction to J</i> Please note the packing charge of £3 per order.
IBM (APL Products)	APL Version 2 Release 1	poa	Full APL2 System for IBM 370 and 390. Product No. 5688-228. See your IBM Branch Office.
	APL2 Application Env't Ver2 Rel1	poa	Run-time Environment for APL2 Packages (IBM 370 and 390). Product No. 5688-229. See your IBM Branch Office
	APL2 for the RISC System/6000	poa	Product No. 5765-012. See your IBM Branch Office.
	APL2 for the IBM PC (US version)	\$495	Product No. 5799-PG. Part No. 6242936. PRPQ No. RJB411 (in HONE, RJ0411). (from IBM Direct, order by Part Number; from your IBM Branch Office, order by PRPQ Number)
	APL2 for IBM PC (European vers)	poa	Product No. 5604-260. Part No. 38F1753
	TryAPL2	free	Free APL2 for educational/demonstration use. Write to APL Products, specifying diskette size desired.
IBM UK	IBM PC APL2	348	APL2 for the IBM PC. From all IBM dealers, including MicroAPL.
Iverson Software Inc.	APL386	\$30	Sharp APL Release 20 for PC 386, 486 with graphics, and ability to operate under Windows.
	APL/PC	\$30	For PC under DOS
	APLWIN	\$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
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	450	
	APL*PLUS II V 4.0	1395	
Optima	APL*PLUS/PC	369	
	APL*PLUS II	950	
	APL*PLUS II PC Developers Kit	poa	
	Dyalog APL	999	
Soliton Associates	SHARP APL for MVS	poa	for IBM MVS mainframes
	SHARP APL for Unix	poa	for IBM RS/6000 and Sun SPARC
Unware	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/UNIX	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
Active Workspace Ltd	Syndicate Manager	poa	Lloyd's managing agent's syndicate / company accounting system. Stamp & Personal accounts (inc. Run offs)
APL-385	APL-385 FSM-385 DRAW-385 DB-385 GEN-385	50(PC),125(mf)	Including ... Screen development Screen design Relational W.S. Miscellaneous Utilities
The APL Group	Qualed	\$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
	JPLS	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
Cocking/Drury (for VSAPL)	E'EMENTS & SHAREFILE	poa	Component files, quad-functions & nested arrays for VSAPL under VM/CMS & MVS/TSO
	COMPILER	poa	The First APL compiler!
	FILEPRINT	poa	Print APL component files
	FILECONVERT	poa	Converts non-APL files to APL
	FILEMANAGER	poa	Extends APL primitives to database management
	TOOLS + UTILITIES	poa	APL Software development tools
	DATAPORT	poa	Information Centre spreadsheet incorporating data exchange between APL, FOCUS, IFPS, SAS, APL/DI, ADHSLI, Lotus123, Visicalc, Multiplan & DIF
(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
	FMT	poa	Full featured FMT for APL2
	WSDOC	poa	Workspace documentation utilities
	FILEMANAGER	poa	Extends APL primitives to database management
(for PC's)	APL*PLUS PC Tools	275	Utilities including: RAM disk, full screen data entry, menu input, report generation, exception handling and games.
	IRMA Module	90	327x IRMA support.
	FIN & STAT. LIBRARY	250	Financial & Statistical routines
	SPREADSHEET MGR	150	APL-based spreadsheet for APL*PLUS/PC. Cell arithmetic; transfers to ASCII & Lotus
CODEWORK	Helm	poa	Decision Support system for top management. Developed in Italy over 7 years. Requires APL mainframe or APL*PLUS/II
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!
Impetus Ltd	<i>Impetus</i>	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.
Interprocess Systems	IEDIT	\$3000-5000	Full screen APL2 editor with immediate APL execution, and full-screen debugger
	AOC	poa	APL2 Optimizing Compiler, translates APL2 functions to FORTRAN programs.
(mainframe)	AFM	\$6500-15300	High performance component and keyed file system (VS APL and APL2)
(PC)	AFM	\$175	Single user component and keyed files for APL2/PC.
	Enhanced Format	\$2575	A QuadFMT data formatter for VS APL and APL2
	PowerCode	\$2000	External functions for APL2
	CALL/AP	\$4700	For calling non-APL programs (VS APL and APL2)
	WSORG	poa	Full-screen Workspace Organizer for APL2.
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.
MicroAPL	MicroTASK	250	Product development aids
	MicroFILE	250	File utilities and database
	MicroPLOT	250	Graphics for HP plotters etc
	MicroLINK	250	General device communications
	MicroFORM	250	Full screen forms design
	MicroSPAN	250	Comprehensive APL tutor
	MicroPLOT/PC	250	For APL*PLUS/PC product
	MicroSPAN/PC	250	APL self Instruction for APL*PLUS/PC
	STATGRAPHICS Rel 5	590	
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	Consultancy	poa	PC Based APL system design, programming and implementation.
Adfee	Consultancy	poa	Development, maintenance, conversion, migration, documentation, of APL products in all APL environments
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.
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
Cocking/Drury	Consultancy	175-275 275-350 300-450 400-600 450-750	Junior consultant Consultant Senior consultant Principal Consultant Managing consultant
David Crossley	Consultancy	poa	Broad experience in many APL environments
Peter Cyriax	Consultancy	100-150 120-200 160-300	Junior Consultant Consultant Senior Consultant
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
General Software	Consultancy	from 120	
Greymante Assoc Ltd	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.
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.
Kestrel	Consultancy	poa	All APLs, all environments. Design, analysis, coding, maintenance, documentation, training, interfacing.
Lingo Allegro USA	Consultancy	poa	General APL consultancy specializing in Prototyping, Migration, Mainframe to PC Downsizing, Performance Analysis, Troubleshooting, and Graphics.
MicroAPL	Consultancy	poa	Technical & applications consultancy.
Ellis Morgan	Consultancy	250-500	Business Forecasting & APL Systems.
Optima	Consultancy	poa	A range of consultants with 3-15 yrs APL PC and mf experience.
Parallax Systems Inc	Consultancy	\$750	Introductory APL, APL for End-user & Advanced Topics in APL
QB On-Line	Consultancy	350	Specialising in Banking, Financial & Planning Systems.
Rochester Group	Consultancy	poa	Specialise in MIS using Sharp APL
Rex Swain	Consultancy	poa	Independent consultant, 15 years experience. Custom software development & training, PC and/or mainframe.
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
Adlee	Employment	poa	Contractors and permanent employees
APL People	Employment Agency	poa	Employees placed at all levels.
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-9-1, 24 hours. APL educational material is downloadable free. An additional 90 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 Camecho & Ziemann
	APL Programs for the Mathematics Classroom (Springer-Verlag)	16	185pp by Thomson
	J Dictionary	16	by Ken Iverson
	Programming in J	10	75pp by Ken Iverson
	Arithmetic	12	118pp by Ken Iverson
	An Introduction to J	8	47pp by Ken Iverson

	Tangible math	8	36pp by Ken Iverson
	Sharp APL Reference Manual	18	349pp by Berry
	APL Press Books	poa	A comprehensive selection of early APL literature
	<i>Please note there is a packing charge of £3 per order</i>		
IBM (APL Products)	APL2 Keypacs	poa	Keypacs for the PC and PS/2 (USA and UK standard). Product Number SX80-0270.
	APL2 Keyboard Stickers	poa	Product Number SC33-0604.
Iverson Software Inc.	Programming in J	\$15	76pp.
	Tangible Math	\$12	34pp.
	Arithmetic	\$18	123pp.
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.
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

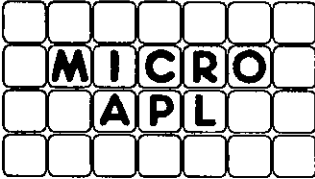
GROUP	LOCATION	JOURNAL	OTHER SERVICES	Ann.Sub.
ACM/SIGAPL	International	Quote Quad		
APL Bay Area	USA N. California	APLBUG	Monthly Meetings (2nd Monday)	\$15
APL Club Austria	Austria	-	Quarterly Meetings	200AS(indiv), 1000AS(corp)
APL Club Germany	Germany	APL Journal	Semi-annual meetings	DM60
Ass. Francophone pour la promotion d'APL	France	Les Nouvelles d'APL		
BACUS	Belgium	APL-CAM	Conferences & Seminars	£18 (\$30)
CPC UG APL SIG (Capital PCUG)	Washington, D.C.	Capital PC Monitor	Monthly meetings, occasional classes	free
Dutch APL Assoc.	Holland	-	Mini-congress, APL ShareWare Initiative	
FinnAPL	Helsinki, Finland	FinnAPL Newsletter	Seminars on APL	100FIM(private), 30(student), 1000 (Co)
SwedAPL	Sweden	SwedAPL Nytt	Semi-annual meetings, seminars	SEK 75
SWAPL	Texas, USA	SWAPL		\$18
Swiss APL (SAUG)	Bern	Part of Qtly SI-Info		SF60 (SI) + SF20 (SAUG)
Sydney APLUG	Sydney, Australia	Epsilon	Monthly Meetings	

VENDOR ADDRESSES

COMPANY	CONTACT	ADDRESS & TELEPHONE No.
ACM/SIGAPL	Donna Baglio	ACM, 1515 Broadway, New York, NY 10036 USA Tel: (212) 626-0606 Email: baglio@acm.org
Active Workspace Ltd	Ross D Ranson	Moulsham Mill Centre, Parkway, Chelmsford, Essex, CM2 7PX. Tel: (0245)-496647; Fax: 0245-496646.
Adfee	Bernard Smoor	Dorpsstraat 50, 4128 BZ Lexmond, Netherlands. Tel: 31-3474-2337, Fax: 31-3474-2342
APL-385	Adrian Smith	Brook House, Gilling East, York. Tel: 04393-385

APLBUG	Lewis H. Robinson	1100 Gough St, Apt 14A, San Francisco, CA 94109, USA Tel: (415) 929-2058
APL Club Austria	Erich Gall	IBM Österreich, Obere Donaustrasse 95, A-1020 Wien, Austria
APL Club Germany	Dieter Lattermann	IBM Germany, Wilckensstrasse 1a, D-6900 Heidelberg, Germany. Tel: (49) 6221-404-243
The APL Group Inc	Stuart Sawabini	644 Danbury Road, WILTON, CT 06897 USA Tel: (203) 762-3933 Fax: (203) 762-2108
APL People / Software	Jill Moss	The Old Malthouse, Clarence St, BATH, UK NS. Tel: 0225-462602
Association Francophone pour la promotion d'APL	Dr. Gérard Langlet	SCM, C.E. Saclay, F-91191-Gif sur Yvette, France. Fax: (33) 1 39 08 79 83
Atlantis Software	Arthur Whitney	1105 Harker Avenue, Palo Alto, CA 94301 USA
BACUS	Joseph de Kerf	Rooienberg 72, B-2570 Duffel, Belgium.
Camacho	Anthony Camacho	11 Auburn Road, Redland, Bristol BS6 6LS. Tel: 0272-730035. email: acamacho@cix.compulink.co.uk Reutemet (Sharp): ACAM
Ray Cannon		21 Woodbridge Rd, Blackwater, Camberley, Surrey GU17 0BS Tel: 0252-874597
Paul Chapman		41 Lambs Conduit Street, London WC1N 3NG. Tel: 071-831-3762.
Cocking & Drury Ltd.	Romilly Cocking	180 Tottenham Court Road, LONDON, W1P 9LE Tel: 071436 9481 Fax: 071-436 0524
CODEWORK	Mauro Guazzo	Corso Cairoli 32, 10123 Torino, Italy. Tel: 011 885 188 Fax: 011 8122-652
CPC UG	Lynne Startz	Capital PC User Group, 51 Monroe Street, Suite PE-2, Rockville, Maryland 20850, USA. Tel: 301-762-9372.
David Crossley		187 Le Tour du Pont, Quaiarier Le Mourre, 84210 ST DIDIER, France Tel: 90-66-08-87
CYBEX AB	Lars Wentzel	Gruvgalan 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. Tel: 071-229-5344
Datatrade Ltd.	Ian Tomlin	1 & 2 Sterling Business Park, Salthouse Road, Brackmills, Northampton, NN4 0EX. Tel: 0804-780241
Dutch APL Association	Bernard Smoor (Sec)	Postbus 1341, 3430BH Nieuwegein. Tel: 03474-2337
Dyadic Systems Ltd.	Peter Donnelly	Riverside View, Basing Road, Old Basing, Basingstoke, Hants RG24 0AL. Tel: 0256-811125 Fax: 0256-811130
E & S Associates	Frank Evans	19 Homesdale Road, Orpington, Kent BR5 1JS. Tel: 0689-824741
FinnAPL		SUOMEN APL-YHDISTYS RY, FinnAPL RF, PL 1005, 00101 Helsinki
General Software Ltd	M.E. Martin	22 Russell Road, Northholt, Middx, UB5 4QS. Tel: 081-864-9537
Greymantle Associates	George MacLeod	Barthum House, Ravens Lane, Berkhamsted, Herts, HP24 2DY Tel: 0442-878065
H.M.W. Trading Systems	Stan Wilkinson	Hamilton House, 1 Temple Avenue, Victoria Embankment, London EC4Y 0HA. Tel: 071-353-4212; Fax: 071-353-3325
HRH Systems	Dick Holt	3802 N Richmond St, Suite 271, Arlington, VA 22207 Tel: (703) 528 7624 Internet: dick.holt@acrn.org
Michael Hughes		28 Rushton Road, Wilbarston, Market Harborough, Leics., LE16 8QL. Tel: 0536-770998
IAC/Human Interfaces	Ian A. Clark	9 Hill End, Frosterley, Bishop Auckland, Co. Durham DL13 2SX Tel: 0388-527190. Email: clark.i@applelink.apple.com
I-APL Ltd	Anthony Camacho (for queries, order forms)	11 Auburn Road, Redland, Bristol BS6 6LS. Tel: 0272-760036 email: acamacho@cix.compulink.co.uk Reutemet (Sharp): ACAM
	J C Business Services (for pre-paid orders only)	56 The Crescent, Milton, Weston-super-Mare, Avon, BS22 8DU
IBM (APL Products)	Nancy Wheeler	APL Products (M46V/D12), IBM Santa Teresa, PO Box 49023, 555 Bailey Avenue, San Jose CA 95161-9023, USA. Tel: 1-408-453-APL2
Impetus Ltd	Cedric Heddle	Rusper, Sandy Lane, Ivy Hatch, SEVENOAKS, Kent TN15 0PD Tel: 0732-885126
INFOSTROY	Alexei Miroshnikov	3 S. Tulenin Lane, St. Petersburg 191186 Russia. Tel:+7 812-3111611 Fax:+7 812-3153321 Email:aim@infostroy.spb.su
Interprocess Systems Inc.	Stella Chamberlain	11650 Alpharetta Highway, Suite 455, Roswell, Georgia 30076, USA Tel: (404) 410-1700. Fax: (404) 410-1773
Intelligent Programs Ltd	Mike Bucknall	9 Gun Wharf, 130 Wapping High St, London E1 9NH Tel: 071-265-1120

Iverson Software Inc.	Eric Iverson	33 Major Street, Toronto, Ontario, Canada M5S 2K9 Tel:(416) 925-6096 Fax: (416) 488-7559
Kestrel Consulting	Mark Harris	Business & Technology Centre, Bessemer Drive, Stevenage, Herts SG1 2DX Tel:0438-310155 Fax:0438-310131 E:kestrel@apl.demon.co.uk
Lingo Allegro USA Inc.	Steven J. Halasz	203 North LaSalle St., Suite 2100, Chicago IL 60601 USA Tel: (312) 556-1342 Fax: (312) 346-9603
Mercia Software Ltd.	Gareth Brentnall	Holt Court North, Heneage Street West, Aston Science Park, Birmingham B7 4AX. Tel: 021-359-5096. Fax: 021-359-0375
MicroAPL Ltd.	David Eastwood	South Bank Technopark, 90 London Road, LONDON SE1 6LN Tel: 071-922 8866 Fax: 071-928 1006
Ellis Morgan		Myrtle Farm, Winchester Road, Stroud, Petersfield, Hants. Tel: 0730-263843
Optima Systems Ltd	Paul Grosvenor	Airport House, Purley Way, Croydon, Surrey CR0 0XY Tel: 081 781-1812 Fax: 081 781-1999
QB On-Line Systems	Phillip Bulmer	5 Surrey House, Portsmouth Rd., Camberley, Surrey, GU15 1LB. Tel: 0276-20789. Fax: 0276-683427. Mobile: 0831-307548
Renalssance Data Systems	Ed Shaw	P.O. Box 20023, Park West Finance Station, New York, NY 10025-1510, U.S.A. Tel: (212)-864-3078
The Rochester Group Inc.	Robert Pullman	50 S.Union St., Rochester NY 14607-1828, U.S.A. Tel: 716-454-4360. Fax:716-454-5430
Shandell Systems Ltd.	Maurice Shanahan	Chiltern House, High Street, Chalfont St. Giles, Bucks., HP8 4QH. Tel: 02407-2027. Fax: 02407-3118
Soliton Associates	Laurie Howard	Soliton Associates Ltd, Groot Blankenberg 53, 1082 AC Amsterdam, Netherlands Tel: +31 20 570 8733 Fax: +31 20 570 8758
Rex Swain		8 South Street, Washington, CT 06793, U.S.A. Tel: 203-868-0131
SWAPL	Stuart Yarus	PO Box 210367, Bedford, Texas 76095, USA Tel: (817) 577 0165
SwedAPL	Glan Medri	Box 16181, S-103 24 Stockholm, Sweden Tel:+46 (8) 96 09 47
Swiss APL User Group		Swiss APL User Group, CH-3001, Bern 1, Switzerland Email: sl@lif.unizh.ch
Sydney APLUG	Rob Hodgkinson	PO Box 1511, Macquarie Centre, NSW 2113, Australia Tel:+612 257 5313
Uniware	Eric Lescasse	15 Rue Erlanger, 75016 Paris, France. Tel: (1) 45-27-20-61. Fax: (1) 45-27-20-71. Telex: 648348F UNIWARE
Wickliffe Computer Ltd	Nick Teller	76 Victoria Rd., Whitehaven, Cumbria, CA28 6JD. Tel: 0946-692588
Warwick University	Prof. Jeff Harrison	Dept of Statistics, University of Warwick, Coventry, CV4 7AL Tel: 0203-523389
Zark Incorporated	Gary A. Bergquist	23 Ketchbrook Lane, Ellington CT 06029, USA. Tel: (203) 872-7806

—	APL Preferences	▽
<h1>APL.68000</h1>		
		
Platforms		
<input checked="" type="checkbox"/> RISC System/6000		
<input checked="" type="checkbox"/> Apple Macintosh		
<input checked="" type="checkbox"/> Commodore Amiga		
<input checked="" type="checkbox"/> Atari ST		
Versions		
<input type="radio"/> Level I		
<input checked="" type="radio"/> Level II		
<h1>APL for GUIs</h1>		

**MicroAPL Ltd., South Bank Technopark, 90 London Road,
LONDON SE1 6LN, UK**

Tel: 071 922 8866 Fax: 071 928 1006
Applelink: microapl Internet: microapl@applelink.apple.com

GDDME - a First Look

reviewed by Adrian Smith

Introduction

GDDME is (as its name implies) a GDDM emulator for Dyalog APL/W. The version I have is an early beta, and is reviewed as such. As it stands, it is technically quite brilliant, but needs quite a deal of polishing to be a usable product. Essentially, it sets out to be a complete emulation of the AP126 calls supported up to GDDM Release 2.0 (roughly 1988): this covers all the standard alphanumeric screen handling and enough graphics to run APL Graphpak successfully. GDDM 3.0 calls, such as the use of partitions, are not supported.

GDDME is written by Andrei Kondrashev, and will be marketed by Lingo Allegro (see Product Guide for availability) at a single-user price of \$1000.

Installation and Documentation

Installation could hardly be easier; GDDME is an independent task, which you can either start from Windows, or kick off with a `⎕CMD` from your APL session. All you need to do is copy `GDDME.EXE` into the directory of your choice, and assign a `PATH` variable to tell APL where to find it. You start the emulation with a function such as:

```

v task←SHARE;rc
[1]  ⍺ Sign on to GDDME using X
[2]  →(2=⎕SVO'X')†0
[3]  ⍺ Set initial value
[4]  X+(,0)''
[5]  rc←'DDE:'⎕SVO'X GDDME'
[6]  task←⎕CMD(PATH,'GDDME.EXE -',⎕WSID,' -GDDME')''
[7]  ⍺ Give it a chance ...
[8]  Wait:→(1≠(⎕SVS'X')[2])/Wait
[9]  ⍺ Interlock
[10] rc←1 ⎕SVC'X'
[11] rc←X
[12] ⍺ Create default page ...
[13] X+(302 0 240 400 0)''
v

```

This opens a small (240 x 400 pixel) window in the bottom corner of your screen, and from now on this window pretends to be a 25 x 80 character window mainframe

screen. The default size is somewhat larger, but for testing it is nice to tuck it away in the corner to leave plenty of screen free for your APL session.

The general quality of the documentation is very high. There are a few minor typos (e.g. the parentheses on line[6] of the above sample were missing), but where it matters — the detailed syntax of the GDDM calls — I have not found any errors at all. The manual is very well produced, and shows clearly the few minor deviations from the standard GDDM syntax. The only big change is that instead of sharing two variables (e.g. *CTLFSM* and *DATFSM*), you only have one (*X* in the sample) which takes both control and data arguments. However all the calls are nicely illustrated with examples, so this should not be a big problem.

Using GDDME for a Simple Screen

Having got over the shock of seeing the Graphpak *SKYSCRAPER* plot for the first time in years, I set to coding up some simple API26 calls to check out the basics. The following little function sets up a two-field format, clears the fields, sets the cursor, and waits for you to type something:

```

V xx;fmt;nmod
[1]  ⍺ Simple GDDME test
[2]  ⍺ Make a simple format and set up screen ...
[3]    fmt←2 5ρ1 4 10 1 24,2 8 10 5 12
[4]    X←(402,(ρfmt),,fmt)''
[5]    X←(407 1 4,407 2 3)''
[6]  ⍺ Fill top field with green underscores ...
[7]    X←(424 1 24)(24ρ'_' )
[8]  ⍺ ... and lower field with red dots
[9]    X←(424 2 60)(60ρ'.' )
[10] ⍺ Set Cursor and Hold it ...
[11]  X←(430 1 1 1)''
[12]  □DL 1 ⍺ Essential!
[13]  X←(,101)'' ⋄ nmod←X ⋄ 'Read result was: '(5+nmod)
[14]  ⍺ Query modified fields ...
[15]  nmod←ϕnmod
[16]  X←(420,nmod)'' ⋄ 'Flds to read: '(5+X)
V

```

The first element of the couplet is always the control variable. It is a minor annoyance that even when the data variable is ignored (as for the format lines 4-5) you still have to supply it. Otherwise, this is all absolutely standard GDDM stuff, and should look quite familiar to anyone who has programmed mainframe screens. The only significant snag is on line[12], where I found it essential to wait a little while before issuing the ASREAD call on line[13]. If I ran the code from the session (or under Trace) — no problem. If I let the function rip through it

flips up the GDDM screen, but returns immediately to APL with a result of 0 0 0 0 0. I suspect a nasty interlock problem on the DDE connection which may only show up on a fast machine when you have little or no APL processing between calls.

Assuming you let it wait, the result looks like:

```

Hello from GDDME_____

Here is some
data in a fi
eld.....
.....
.....

```

The window behaves just like a mainframe screen, but you can position the mouse with the cursor, and you don't need to <reset> if you accidentally type something in the wrong place. When you press <enter> or a function key, it returns with the customary information about what you hit, and how many fields have been modified.

Things that Need Fixing

Programming this beast is rather like riding a monocycle up an alpine pass with no safety fence! It is very sensitive to abuse, and I quickly learned to be extra careful — the alternative is frequent use of Ctrl-Alt-Del to escape from a totally locked-up machine. Specifically:

- it is not a good idea to close its window with the control menu. This kills the task OK, but if you attempt to re-start with *SHARE* the shared variable fails to couple and you can't type anything in your Dyalog session ever again!
- the field behaviour in <Insert> mode is distinctly weird, and definitely not standard GDDM behaviour, although in most instances it is harmless.
- the function key mapping is strange. I think most mainframers expect F11 and F12 to behave as advertised, and Shift+F1 to give F13. This is not what it does!

- during an ASREAD it locks the mouse totally into the client area of the GDDME window — you can't even get at its own title-bar or system menu. This is quite unacceptable behaviour (Windows is a multi-tasking environment after all), and also dangerous. On a couple of occasions I managed to kill the GDDME task, but my mouse was still trapped in the rectangle where its window had been!
- similarly, during an ASREAD, none of the standard Windows switching keys work. This prevents you from doing basic and obvious things like snapping a copy of the window to the clipboard. You can't even get at its own control menu to select the 'Print' option (see the mouse problem above).

Suggestions for Improvement

As I said at the beginning, the emulation is a technical tour-de-force, but the interface could do with some tidying. Specifically:

- it insists on a white background. This is probably the worst choice, as all the foreground colours are bright. Our mainframe screens are heavily into cyan, yellow and pink; all these colours are virtually invisible on a white background. I use the IRMALAN 3270 emulator on the rare occasions I need the mainframe — this defaults to black (acceptable) but allows you to choose dark grey which I find much more restful.
- Courier may not be the only monospace font available, and again the user should be offered a choice. It may be worth abandoning the total scalability of the window and using the font set that Windows installs for its DOS boxes. These are very much more readable, particularly when running at small sizes.
- simple clipboard support would be nice, even if only at the level of a system-menu option to copy the whole text screen to the clipboard.
- trapping the mouse really is not on! The user spends 99.98% of his time at an ASREAD; it is quite ludicrous to prevent him from moving / resizing / minimising GDDME, let alone switching out to do something else.

Summary

This product shows a lot of promise. I can think of a number of major mainframe applications I could try out on it with minimal effort. If you have an urgent need to get your GDDM code off a mainframe, give Lingo Allegro a call and push them for a date on a pukka version. I am pretty confident that internally this thing is rock-solid; it just needs to be made more approachable.

Helm — A Company-oriented Decision Support System (DSS)

reviewed by Jon Sandles

Introduction

Vector received the installation disks for Helm in April 1993 — asking if it could be reviewed. It is written in APL by a company called Codework from Italy — potentially available for TSO and MS-DOS 286/386/486 and also UNIX. Apparently it is used by about 100 installations — including FIAT. The particular version we received ran under the APL*PLUS II development interpreter and hence required a dongle. Of course, if you do not already have a dongle this makes it quite an expensive piece of software as you will have to splash out for one in the first place. Helm believe there are good reasons why you need to be in a non-runtime environment for this type of system. (They have tried a runtime version but found it difficult to comprehensively trap all the errors in a suitable way — this type of system will inherently produce errors because the user types the instructions in.)

Installation

The installation batch file copied a set of packed files/workspaces into the `c:\helm` directory. The batch file `helm` merely calls the non-runtime interpreter from this directory. But, you'll need to put the path of your APL interpreter in here and make sure it picks up the file `config.apl` in the `c:\helm` directory which then loads the correct initial workspace etc. This bit is not very well documented, and if you had never used PLUS II before you could well struggle. Even though I made it that far I still got thrown out due to a reference to a missing file `aplkeys.apl` in the file `config.apl`. I commented this line out and finally got in. (None of the F-keys then worked — Codework say they left the file out at the last minute for copyright reasons.)

Once you are in you have to follow an installation procedure — firstly declaring your hardware configuration and then registering your name and company details. This produces a 'plate no' which you have to ring Codework (in Italy) with to complete registration. They then take the codeno and name you have typed in and calculate the password you should enter to complete registration and allow you to use the rest of the code. (When I rang Codework they seemed surprised that I had made it that far!!) After putting in the password that Codework supplied I was finally in.

Initial Impressions

I initially set up the software in 'semi-automatic' mode — this is basically a set of menus which allow you to pick an option which then loads the relevant workspaces. The actual screens are just interactive text screens which prompt you for function names/arguments. You can create/edit your initial data by using spreadsheet style data entry screens. You can then 'analyse' the data by selecting the dimensions of the data which you are interested in (seemingly limited to 3 dimensions) and then performing the appropriate functions. There are full-screen graphics which have some useful features. Difficulties I had early on were that I sometimes found problems in relating what I had in the documentation to what I found on screen. I could not send my prints to a networked printer (although I am told there is a way). There was no mouse support (apparently this is in the development version), I could not add a new database of my own (it came with a couple of example databases) — due to not having any F-keys. (I eventually found a way around this as well.) It should be noted that I did not have the full system documentation — I just had what was presumably a precis of the system's functionality.

The overall feel of the system is like the old mainframe data/table management systems. It is typical of its genre in the way that the user must type in the commands he wants to run to get the results he requires. Typing errors are usually rewarded with an irrelevant error message (to the user that is) but if you know the commands available the rewards are great. A typical session starts out with (user commands are in italics):

```

+-----+
|  SIZE      SUMMARY      |  QUERY      DRAFTPRINT  PAGEPRINT  |
|  LOCATE    PROCEDURES   |  PLOT        PROFILE    CLUSTER    |
|  SESSION   LABELS       |  HISTOGRAM   SORTPRINT   MISSING    |
|  QUIT      |  FRAMEPRINT  BOXPRINT   EXPORT     |
|              |  BROWSE     |
+-----+-----+
|  SELECT    DROP         |  COMPUTE     PERCENT     GROWTHRATE |
|  RESET     |  MEAN        TODATE     AGGREGATE   |
+-----+-----+

Enter your choice :
SUMMARY
Original data base : MINI
Data cube is in memory 1000 bytes
Available memory 1373820 bytes

Size of the current data cube :
  5  ACCOUNTS
  5  YEARS
  5  BANKS

etc.....

```

Commands like COMPUTE, SELECT etc. can be issued on their own and you are then prompted for the types of arguments they can take. If you know the full command you can type it in full and get the result immediately. This may appear at first sight tedious, but it is the flexibility which this *modus operandi* supports which makes it so popular and successful. The initial effort required to learn the ins and outs of the command language is high compared to the reward gained (especially compared to the relative ease with which you can get large rewards from recent Windows software). But, over time the command-driven approach of systems like Helm require much less effort and gain much more reward — and more importantly they are able to cope with the unpredictable nature of ad hoc applications that DSSs are often asked to tackle. The menu-driven approach often hits a brick wall which cannot be overcome without considerable redevelopment.

This flexibility is why these systems are still popular. The menu-driven approach is beginning to hit back by incorporating macro-languages and user-defined toolbars to enable the user to tackle a wider variety of problem within a single system. But because users have become used to the flexibility of their old command-driven systems many are reluctant to change.

Speaking to Helm (in English I'm afraid!) you soon get the same old picture that is besetting most APL system developers: the system was developed on Mainframe TSO and then when the users started turning their mainframes off and turning their PCs on, it was ported to a PC environment, but of course they wanted it to look the same. Hence the current look and feel of the system. Now of course the users are turning round again and asking for a Windows environment. (Helm are currently developing in Dyalog APL — but are sceptical about the Windows environment — who can blame them — it's going to be a lot harder to maintain the system across all the different platforms I listed in the first paragraph.)

General Reflections

Why would anybody want Helm? To begin with I was not sure. In the world of Windows most of what this does is an awful lot easier in tools like Excel etc. Text-based interaction went out with the ark (well at least a year ago) — but considering it is *supposed* to look and feel like a mainframe system it is quite good. The prompting and help is reasonable and the manual is full of examples. The data is entered in three dimensions via screens very similar to Lotus (or from text file etc.). Helm believe all data comes in three dimensions — but there are ways to have more or less. The system works best with three.

Here are a few example screens which should give you an idea. First the natural language interface:

```

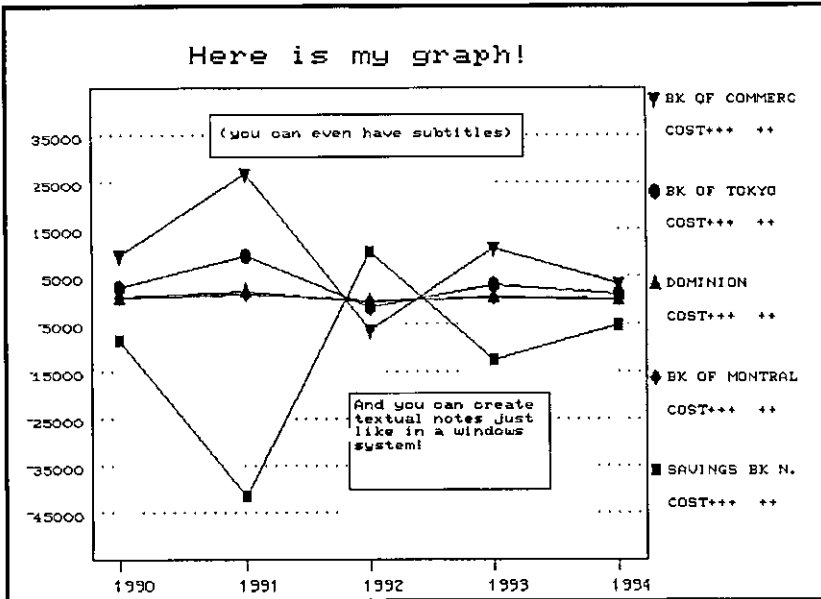
Enter your choice:
QUERY
This module accepts natural-language queries relating to the current
data cube.
See the manual for a comprehensive definition of queries.
You can also ad-lib simple queries on the basis of these examples :

A query can contain both these keywords
average total max min how many
label of those which dont have
equal above below from to and or not
and also ACCOUNTS enclosed between the two delimiters
<...> (See function key F6).
Sample queries :

average <PROFIT AFTER TAX> of those which have <GROSS PROFIT> above 100000
label of those which dont have <COST> from 100 to 300
how many have <COST> above 80000
<PROFIT AFTER TAX> of those which have <GROSS PROFIT> above 500000

Enter your query : etc...
  
```

At the end of the day the system is used to produce reports — and so the graphical outputs are quite important — here is a sample graph:



I could have done with the full manual to determine how good this interface language was, but it appeared from my limited efforts that it was quite effective (it was tempting to put SQL-like syntax in which did not seem to work).

Of course it's all in colour if your hardware supports it! Without the manual I could not establish how many different sorts of graphs you could do. (It appeared to be limited to line plots and histograms.) But the graphs I tried were very flexible in terms of layout and titles etc. — which is never easy to do effectively in this type of environment.

Conclusion

It would be unfair for me to suggest that there is anything wrong with this system. Helm have successfully marketed this system initially onto mainframes and then ported it when required to an environment that their users felt comfortable in. In five years time when the new environment comes along they will port the system again and the users will still be able to access their old data and have the same functionality. (Whereas everyone else will be splashing out for the new Excels and Words of the new world and typing all the data in for the third time.)

Small software vendors like Helm need to be applauded for supplying good usable systems to companies that require more than what is commercially available at the time, and when the commercial sector catches up they are still able to compete. And it's all done in good old APL!





The Skydome and CN Tower
from Wards Island



Jim Brown watching
Dmitri with Jim's guitar



A Demo of K
Keith Smillie, Jim Lucas, Anthony Camacho,
with Arthur Whitney and Jacob Brickman

APL 93 AT TORONTO



CONFERENCE ROUNDUP

Thanks to everyone on the Vector working group for prompt and thorough notes on the tutorials and presentations.

Particular thanks to Richard Procter, Dieter Lattermann, Marc Griffiths, Diane Whitehouse, Ray Cannon, John Searle, Phil Benkhard, Anthony and Sylvia Camacho and Carlton Moore for an excellent set of photographs — the usual apologies for not fitting more of them into the available space.

The Lessons of Toronto APL93: Pain and Painkiller and well worth "Taking a Closer Look"

by *Sylvia Camacho*

I was introduced to APL in 1980, after being connected with the computer business for over 20 years. My first reaction to APL was one of simple delight and the expectation that, as computing hardware increased in power, this astonishing language would take the programming profession by storm. I get no marks for foresight but as I listened to the papers at APL93 I was working hard on my hindsight.

The Pain

Donald McIntyre '*An Introduction to J*': the reaction of the APL community to the introduction of J has been instructive. One old APL hand expressed to me in private the feeling that Ken Iverson had betrayed his loyal followers. This led me to reflect that while APL is above all things an intellectual pursuit, one does not have to work in the field for long before becoming aware that it generates pain.



Roger Hui and Donald McIntyre

Fundamental change to a paradigm on which hard thought has been expended is very difficult to accept and persuasion to attempt it generates strong emotions. For years APL has provoked this pain in computing, mathematical and scientific communities outside the APL pale but J was born within the pale and is pricking us all.

Nevertheless this Workshop was over-subscribed and enthusiastically received despite being conducted in a high humidity heatwave in a room of 40 people and 30 computers with inadequate air conditioning. Donald

makes no secret of the fact that he found J hard to come to terms with after years of APL. I have now attended his workshop twice and I have not attempted any J,

notwithstanding which I am convinced Ken Iverson has surpassed APL in power and elegance. One of the most striking differences is that whereas APL borrows from and extends the use of mathematical terms, J borrows from and extends the use of grammatical terms. Why is this an advance? I can only say that the convergence of mathematical and natural language in the terminology used to describe the J structure 'feels right' and that this is a good scientific reason for adopting it: ask any leading physicist.

Donald McIntyre: *'The Triumph of Symbols over Words'*. This was the opening plenary session. The theme has been explored before by Donald McIntyre although never before to the skirl of the pipes. The fact that J has abandoned the special APL symbols gives the theme an added significance. The APL community has grown used to the arguments accompanying its challenge to conventional mathematical notation. Now APL itself is challenged by the expression of similar ideas, extended and more coherent, but in the limited and arbitrary notation of ASCII. Ah, the pain of it!

Donald's genius is to demonstrate amusingly that this conflict is as old as mankind. In particular Egyptian hieroglyphs changed from ritual priestly pictures to something more secular and with phonetic overtones. This being the language of the gods the apparently useful transition was resisted by the priestly establishment and the Egyptian language came to be written, for secular purposes, in the Greek notation. Thus the point is made that the notation may be better or worse for many reasons but it must not be confused with the language itself or the ideas the language expresses.

Is the pain eased by knowing that it was felt in all previous articulate generations? Perhaps not but it makes a superb lecture.



Part of the Plenary Audience

The Toiler in the Elysian Fields

G rard Langlet *'Building the APL Atlas of Natural Shapes'*: the most notable thing to be said of G rard is that he does not feel the pain of the paradigm shift because he takes the tools Iverson gives him and builds a world of his own. Take, he says, a minimum set of numbers/nouns and functions/verbs and operators/adverbs: what does the name matter. Take, especially, not-equal scan and binary vectors and build models. I will show you, he says, that this simple set of tools can model the most profound theories in physics and biology (last year's paper) and in fractals and related non-continuous, non-linear mathematics (this year's paper). His bravado has an intoxicating effect and his pictures are beautiful and some are in the Software Exchange.



G rard Langlet

Norman Thomson *'Understanding ANOVA the APL Way'*: Norman's paper followed on from a typical polymath offering by G rard Langlet. The two papers formed an interesting contrast but, as Norman said in his opening remarks, they both have an obsession: G rard's obsession is not-equal scan, Norman's obsession is enclose with axis. He discovered that this was the key to developing functions to be used for the set of statistical procedures known as Analysis of Variance.

The Painkiller

A different aspect of the pain associated with APL is generated by the tension between the detached intellectuals and the professional programmers with employers to satisfy and a living to earn. This Conference marked the start of the relief of that tension. The work put in by the APL Vendors is bringing Graphical User Interfaces to the market very shortly after the release of MS Windows 3.1. This Operating System is achieving the wide and deep penetration needed to set a de facto standard.

Adrian Smith '*Co-operative Programming with Windows DDE*': with the advent of DDE (Dynamic Data Exchange) and OLE (Object Linking and Embedding) and the, as yet something less than ideal, availability of DLLs (Dynamic Link Libraries), APL or J can be exploited to the full, doing the hard algorithms and leaving the hardware-user interaction where it belongs, in the Operating System.



Adrian Smith and Dieter Lattermann

Adrian has adapted to this almost pain-free world with enthusiasm and demonstrated how to exchange data with other applications running under Windows to display a graph, format a document or manage a database. Excel is an excellent spreadsheet but it cannot invert a matrix? Then why not set it up with a new toolbar with controls identified by any picture you fancy (even the domino from that controversial notation). Then you can call APL from Excel and knock the numbers into shape. Whether it be with APL, J, Excel, Word or Access let each cobbler stick by his last and whatever the shape of the users' feet their shoes will not pinch. Thus we can expect that next year's Conference will be painless for both APL and J practitioners.

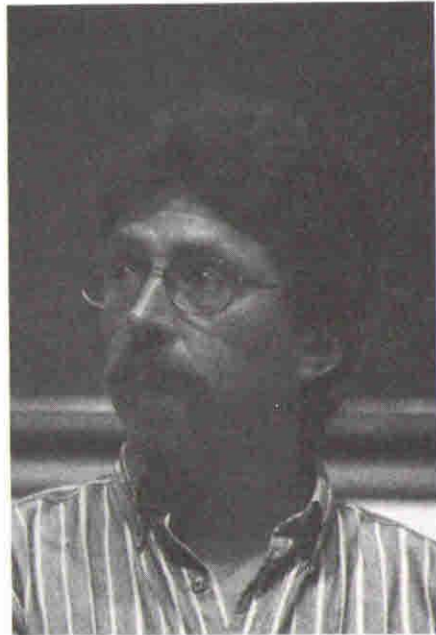
Eric Iverson '*Windows GUI Programming in ISIAPL*': Eric's Workshop was one of the highlights of the Conference for me. I have been attending APL conferences since 1986 and at each I have heard pleas from APL programmers to APL implementors for tools to enable them to compete with the growing number of packages featuring Graphical User Interfaces. The various vendors of APL interpreters attempted to respond with Auxiliary Processors and then with system functions like `□WIN` but such language extensions are expensive to produce and expertise gained in one is not transferable to a different vendor's product. Last year, however, Microsoft moved the game into an entirely different ballpark. The 3.1 version of their Windows Operating System sets a PC standard for GUI at an affordable price and provides the framework within which APL implementors can develop Windows tools quickly and cheaply. The May 1993 release of Iverson Software APLIWIN now goes a very long way towards responding to the clamour for tools to enable APL applications to compete in

look and feel with the so-called Fourth Generation languages while having all the power of an advanced APL including nested arrays and complex numbers.

Eric began by explaining that he admires the Microsoft Visual Basic approach but that he feels when using it that he 'does not know what is going on under the hood'. Therefore the goal for his APL and J Windows implementations was to create a *Window Driver* ($\square WD$) which would be easy to learn and to use for straightforward applications. He envisages the Window Driver itself as a learning tool and so it takes the form of a script language like PostScript or SQL. He hopes to make the Window Driver itself available as a DLL (Dynamic Link Library) and thus able to be used in conjunction with other products, even other APL implementations. APLIWIN will also run under the OS2 Windows emulator.

One of the most attractive APLIWIN features is an editor (*w \square vedit*) which makes the creation and customisation of Windows objects easy and fast. Multiple boxes and buttons can be created and aligned and the window can be instructed to resize automatically to accommodate them.

APLIWIN has good on-line Help which should be printed to constitute a windows facilities manual. It comes with a set of workspaces which demonstrate GUI code and provide tools for the application builder. The price is very good: \$30 US for APLIWIN, \$30 for the language manual and free run time (£25 each from I-APL in UK). The language manual (fundamentally Sharp APL, of course) has the typical Iverson terseness and would be improved by an index but I have used APLIWIN for real (i.e. paid!) work despite having been brought up on APL*PLUS.



Eric Iverson

The APLIWIN system function is $\square wd$ with a string argument consisting of elements of the scripting language with the semi-colon separators: e.g. $\square WD$ ' *ptop*; ' keeps all the window visible (the colour of the title bar shows whether

it is the window in use). See Adrian Smith's report on the ISI tutorial (page 67) for more details and examples of `□WD` in use.

APLIWIN supports both DDE (Dynamic Data Exchange) and OLE (Object Linking and Embedding). The best way to learn about this is to invoke a second APL session '`winexec "PATH apl.exe"`' and move the new APL window to the lower half of the screen. If, like me, you give APL all the space you can spare you will have to cut the sessions down to a size compatible with simultaneous running. From the top window (the server) now enter `wd 'ddename serv;'` Then `□+d+ '*data' wdg wd'wait;'` puts the upper session into Running mode.

In the lower session (the client) now enter

```
    )load isidemo
    wd 'ddepoke servtopic item "how now cow"'
```

after which 'how now cow' will appear in the top session and both sessions will be in Ready mode.

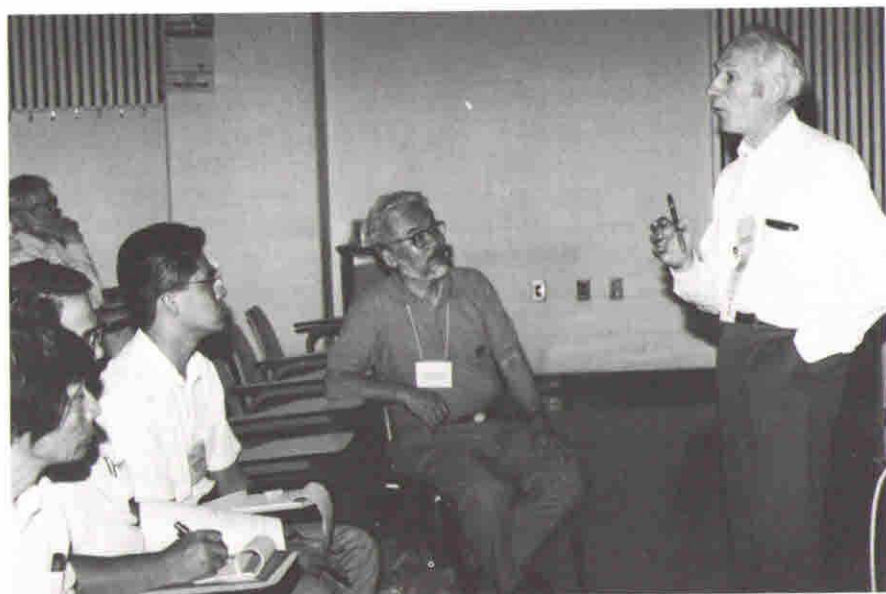
Two APL sessions are useful for learning DDE but one can then progress to making the top session Excel. In such cases the best results are obtained by creating an EXCEL macro to handle the data. The same applies when sharing data with the MS Access relational database.

Conclusion

My conclusion is that complaints that APL applications cannot be given a competitive GUI 'look and feel' are now completely misplaced. There is a very powerful choice between using stand-alone APL which has been given a GUI or using DDE and OLE techniques to exploit the best features of other software.

For largely numeric data capture why try to re-invent the wheel? Use Excel, which is an excellent spreadsheet, and pass the captured data to APL for complex processing. For a good relational database with all the standard facilities use MS Access but pick up APL when the data processing gets tough. For a beautiful document incorporating APL use MS Word and cut and paste from an APL session.

Now who needs C or Fortran?



Saigusa translates at Donald McIntyre's Japanese Tutorial



Dick Bowman and Larry Moore
Chairman SigAPL and Chairman APL 93



Don Farrand and John McPherson

TUTORIALS

Solving Wicked Problems with APL by Chris Lee, Manugistics Inc.

reported by Jonathan Barman

Chris made this tutorial thoroughly enjoyable by including masses of the APL array type jokes. I'm afraid not many were recorded. The objective of the tutorial was to demonstrate that APL is an excellent language for solving those difficult problems that give computing departments such pain. Having attempted to do a similar selling job myself, which I found very difficult, I was impressed with the way Chris went about the task. I would love to have had in the audience some of the die-hards that I have tried to convert to see how they would have taken it all.

The first hour was spent defining what Chris meant by *wicked* problems. He poked fun at the advertising for packages which claim that you can do anything without writing a line of code. I liked the set of challenges which included getting your database to dial a phone number when you click on it — without programming. Chris makes the point that if Vendors make packages so simple that idiots can use them, then it is inevitable that sooner or later idiots will be specially hired by companies to use the packages.

Four kinds of programmers were identified. The *Professional Programmer* has a computer science training and solves other people's problems, but has to have the problem described in detail as he has no experience in the field. The *Domain Expert* has a specialist skill, for example Finance or Engineering, and uses computers as a tool to solve his problems. The *Local Guru* is a Domain Expert who has taught himself computing, and has become so expert that he can respond quickly to users' needs because he understands both the problem to be solved and the computing that is necessary. The *Commercial Software Developer* is probably a Local Guru who has identified a market for a general solution to the class of problems in the field of his original expertise. It was interesting to have

these definitions of types of programmers so clearly explained as in the past I have filled all these roles at one time or another.

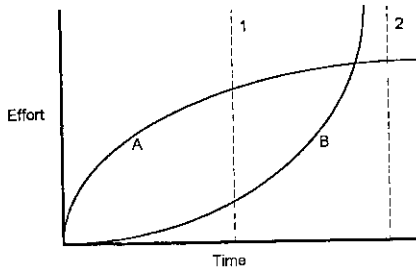
A *wicked* problem is defined as one where there has never been a computer solution applied before, where the definition of the need keeps changing, and anyhow the users are not entirely sure of what they want. The traditional computing solution is to apply the Waterfall Method, which involves the examination of the existing manual system, mapping out the tasks and data flow, and then designing and building a computer system which emulates the manual system. The Waterfall Method is attractive as it makes it clear what has to be done, there are milestones and deliverables, and there is something to manage. This is fine for the big traditional computing problems, but Wicked problems make trouble. They do not allow a proper statement of requirements as the requirements keep changing, which gives the Waterfall Method severe indigestion. This is a recipe for the final disaster when the user says "It's exactly what I asked for ... but it's not what I want".

A solution to the problem is what Chris described as Software Discovery, or Software Sculpturing. A sculptor models a big work in clay before cutting stone. Clay is cheap, bits can be added and moulded easily, and bits can be sliced off and thrown away. Similarly with Software Sculpturing where you think a bit, build a bit, test a bit, tweak a bit and then repeat the process until you get what you want. The main thing is that the process must be cheap and fast. Also, software has to evolve to meet today's need and to stay adaptable. There is a constant requirement for software to grow, change, improve and eventually to be replaced when the original starts looking like a second-hand car with dents.

What tools should be selected for the solution of Wicked Problems? There is familiarity, where the computing department know that they have had a success with a language. If there is a large pool of programmers already using the tool then it should be easy to get good people. The tool must seem to be easy to use. The actual tool chosen will also depend on perceptions, such as the status of the vendor and the quality of the selling.

Programming needs to be carried out in short steps. Each step needs to be able to fit within the short-term memory window, which is about 5 minutes for most people. The bigger the task that can be fitted into the window the faster the overall job gets done. High-level languages allow one to take giant steps. It must be possible to customise the giant steps so that any detailed task can be carried out, and the tool selected must enable easy experimentation and backtracking.

The dying fish diagram illustrates the problem of the different learning curves of the various tools available.



If you have a small job to do which gets you to time point 1 then package B with an easy learning curve is best. However, if you need to do serious programming it is best to master package A as you can never get to point 2 with package B. Wicked problems have a nasty habit of making you think that point 1 is sufficient, but when you get there you realise that you actually need to go to point 2. Sometime after the intersection of the two curves it becomes apparent that package B is not going to do the job and you have to backtrack to get to grips with package A.

Everything in the tutorial up to this point is pretty much standard material for the build-up to the sales pitch for any computing tool that you care to name. Chris's approach to selling APL started with the fact that arrays are the natural way in which to store data. APL's multi-dimensional and nested shapes allow iteration through individual items in an easy and intuitive way. The APL-style instructions on making an omelette would be "Get 6 eggs from the fridge and break them in this bowl". The COBOL instructions would be "Get an egg from the fridge; break the egg in the bowl; have you got 6 eggs yet? no, so get an egg from the fridge; break the egg in the bowl;".

The next step was an attack on the Object Oriented approach. Chris gave a rather nice analogy between OOP and APL by explaining the difference between baobab trees and maple trees. The branches of a baobab tree bend over and the tips plant themselves and produce daughter trees. Over the years you get a small forest of concentric circles with a giant tree in the middle. Maple trees produce copious seeds which spring up over a wide area to produce copies of the original

tree. If there is a local problem, such as a drought, then the baobab tree is killed, but the maple trees are so widely distributed that a local drought will not kill all of them. In the OOP world if there is a problem with the central core of the object hierarchy then much of the dependent code may have to be scrapped. Whilst I agree that there is a difficulty in designing OOP classes, the level of code re-use seems much better than APL. For some reason there is a particular problem with APLers who seem to feel that it is beneath their dignity to re-use anyone else's code, whereas there seems to be less inhibition in the Smalltalk and C++ world.

The importance of being able to experiment cheaply was emphasised. Behind every real success lie a hundred failures, so experiment must be cheap. On the other hand, it is important to avoid spiralling out of control into a black hole. You have to keep track of time and cost. Track this small step and plan the next step towards the final goal. A border needs to be drawn around the definition of the task so that everyone knows when it has been completed. Each step is the waterfall method in miniature. Chris gave us an amusing story of when he was first employed in the computing industry where his job was to keep the progress records of a large 5-year project. Some time before delivery day his system showed that they were 20% behind the target, but all the developers swore that they could catch up. It was not until the actual delivery day that it was announced that they would one month late. In fact they were 3 months late. Whatever language or method you use, you need to know if you are going to be late, and you need to warn the customer as early as possible.

Chris had to rush through the rest of his presentation as we ran out of time. As this was the part of the presentation where he started explaining APL in detail the omission was not too serious. I got a copy of the slides and found that we had seen 34 out of the 78 slides available. The APL exposition in the remaining slides looked comprehensive and well thought out.

In conclusion the benefits of using an Interactive Array-Oriented Environment were given as:

- Dramatically increases the speed of application development
- Streamlines data design to fit requirements
- Greatly simplifies enhancement and modification of applications
- Reduces bugs and time spent de-bugging
- Makes possible the construction of highly complex systems

But we would all say that, wouldn't we?

Using the ISIAPL/W GUI — Tutorial

given by Eric Iverson (detailed notes by Adrian Smith)

Prologue

"Visual Basic: I never had any sense that I knew what was going on — it was 'neat' but obscuring"

Design goals for a common window driver for APL and J:

- easy to learn and easy to use for straightforward "serious" applications
- no intention of allowing the flash whizzbang stuff!
- will be available as a DLL for any Windows product — including other APLs!
- should provide anyone with a simple and fun way of learning Windows programming for just \$30.
- APL keyboard based on the Sharp Union keyboard, with minimal adaptations to conform to Windows conventions.
- the tutorial is all in the online help, and can be printed from a .WRI file which is also on the disk.
- what you do is what you get — there is less magic this way (unlike VB)

"We have the APL character-set; we just don't have an APL programmer" (on making a couple of excusable typing errors)

Here we go ...

```

)clear
clear ws
  ⎕wd 'pc abc;' ⍝ parent create
*type 2 nowait*parentsynowait
  ⎕wd 'pshow;' ⍝ makes window visible
*type 2 nowait*parentsynowait
  ⎕wd 'ptop;' ⍝ so it stays visible!
*type 2 nowait*parentsynowait
  ⎕wd 'reset;' ⍝ destroy all wdws and tidy up.

```

This sequence shows the basic philosophy and syntax of the window driver `⎕wd`. The rule is to define all the details of the form in advance with a 'parent create' before exposing it to view. Note the semi-colons which are mandatory statement

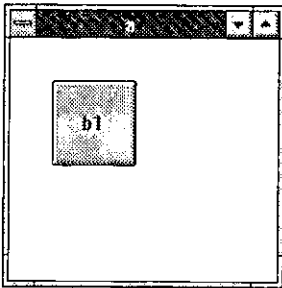
delimiters. To make life a little easier, Eric now loaded a simple utility workspace to cover the `wd` calls, and to reformat the results into a more congenial nested form.

```
)load isiwim
  ed x
  ▽ x ω
[1] -wd ω
  ▽

  x 'pc a;ptop;pshow;'
  x 'xywh 15 15 30 30;'
```

Rectangles are a crucial building block in the Window Driver's world. Here we define a rectangular patch at xy (15,15) (in Microsoft proportional co-ordinates, where (0,0) is the top left corner of the screen) which is 30 units wide and 20 high.

```
x 'cc b1 button;'
```



This uses the current rectangle as the frame for the new object (`cc` = child create). The easy way to write programs of a more useful size is to write the entire `WD` script in advance as a text variable, and then pass the script to `wd` for execution.

```
wp+''
ed wp

rem This is Windows stuff;
pc a;      rem more remarks here if you like;
xywh 15 15 30 20;
cc b1 button; pshow;

x wp ... runs it!
```

Of course just because it was made from a script doesn't stop us adding some extra stuff dynamically:

```
x 'xywh 100 15 30 20;'  
x 'cc b2 button;'
```

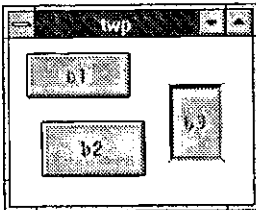
... you don't see anything yet?! That's because it's not inside the visible window ... let's stretch it a bit as you can see it was there all the time! This could be a real nuisance (invisible objects) so we can avoid the problem by adding a 'pas 10 10;' to the program; this is the 'parent autosize' command which ensures the parent is big enough to hold all defined children. Other useful tricks are 'pcenter;' to centre a dialog box on the screen and `ed ''` which re-invokes the editor on the last object. There are also several special parents for particular jobs ... try 'pcd msgthing;' which is Parent Create Dialogbox ... i.e. a simple centred window with a thick border, no resize etc.

Visual Editing

```
wp+''  
ed ''      a whatever we did last  
wp  
pc twp;  
xywh 20 20 20 20;    ... anywhere, any size  
cc b1 button;  
xywh 20 20 20 20;  
cc b2 button;  
xywh 20 20 20 20;  
cc b3 button;    .... may look silly, but wait!  
pas 10 10; pshow;
```

We have the objects we want, so now let's put them in the right place ...

```
t1wp+wdvedit wp ... invoke visual editor
```

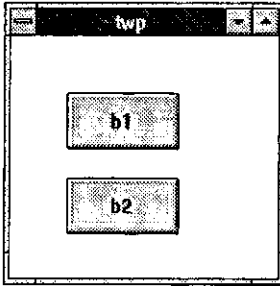


... note that dragging a child causes an automatic resize of the parent. Sooner or later, you have to say "this is aesthetically pleasing" and hit F10 to save the new script. What the editor has actually done is moved the rectangles in the script.

Usually it is more convenient to define one object with respect to another ...

```
pc twp;
xywh 20 20 40 20;
cc b1 button;
adjh 10;    rem bump xywh +10 units downwards;
cc b2 button;
pshow;
```

```
t2wp+ wdedit t1wp
```



You can't grab the b2 button any more, because it has no rectangle! When we move or resize b1 all the others jump to match! Note that you are always moving the rectangles, not the objects which you placed in them.

Moving on

The next thing we need to make is an input area, again this goes within a defined rectangle. As before, you can fix it up with `wdedit` and run it:

```
qq
```

```
pc twp;
xywh 9 57 33 17;
cc OK button;
xywh 11 13 72 13;
cc e1 edit ws_border
```

```
x qq
```

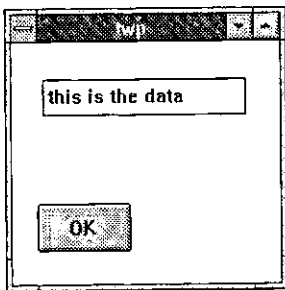
The input box is defined by its type 'edit' (in the example) and by its style 'ws_border' (puts a black frame round it so the user can see where to type).

OK ... we've got these guys, so let's have them talk to us! All you really need is the ability of your APL or J application to wait.

```
wd 'wait;'
```

... a couple of interesting things happen:

- APL is 'running' but it isn't listening to you! You can't type anything in the session.
- you can type into the edit box on the form, and when you hit b2 you get a result back from wd:



```
wd 'wait;'  
*type      5 button  
*parent    twp  
*id        OK  
*pinf     1 0 0 200 200 190 171 1024 768  
e1        this is the data
```

So now we have everything we need to create an application. This can be driven by the simplest of message loops; it can check for a `sysclose` event (the user has killed off the form), else it loops back and keeps listening with another wait.

Q: who decides what events you intercede on?

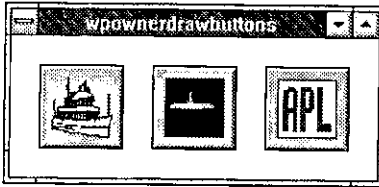
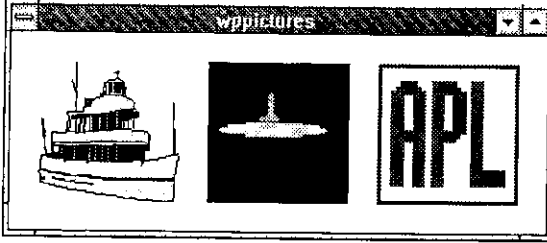
A: I decide! There is no filter which the user can adjust. The wait will terminate on most "reasonable" events, but the list does not include (for example) 'mouse-move'.

More Examples

```

)load isiwip      a examples
saved 1993-05-22 20:57:47
run ''           a runs scripts in the ws

```



This illustrates the use of Windows Metafiles, Bitmaps and Icons, which can be written either to graphics areas or on to buttons. You can also easily access the 'common dialog boxes':

- the standard 'filebox' with the ability to predefine filters and other styles
- the windows 'fontbox' which offers a standard way for the user to make font selections
- the colour-selection box. This offers a colour palette, and returns an RGB triple.
- the standard set of 'msgbox' objects for warnings, queries, critical errors and so on.

After Coffee ...

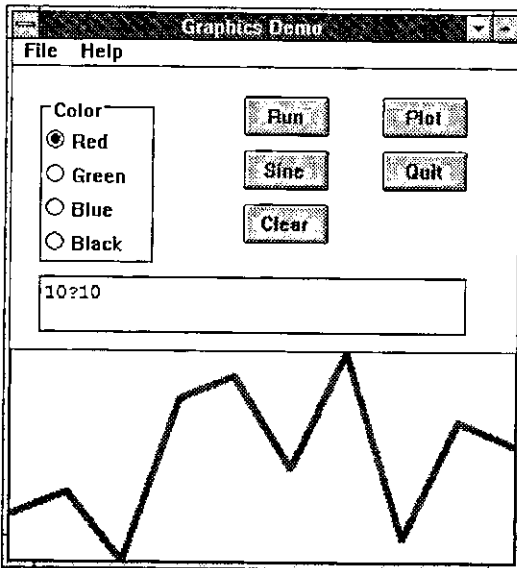
```

)load isidemo
gfx ''  A demo charting function

```

... as someone learning the system, you would play with the application — then take it apart.

- 10 710 ... you get a typical business graph, and you could take decisions based on it:



- this is "sort of" an object, so you can use an invisible edit box to store state information.

```

)load isigen      A skeleton application
gen A runs it as it stands
appcreate 'nap'   A evolve it into our own application
Can't
)wsid nap        A safety feature!
appcreate 'nap'   A now it will work
nap A an exact clone of the skeleton

```

To add a button called 'New' we first edit the main script (called by convention 'napwp') and add a 'cc1 new button;' for 'create child inherit', i.e. make a new button just like the last one. When we run the form and push the button, the application displays "New isn't defined" (because we haven't told it what to do

with the button-press event). The next step is to fix *napmain* to insert some useful code at label 'New:' (e.g. to display the system timestamp in the existing edit field).

This most basic application can be controlled from a very simple message loop — things get harder when there are several active forms as it is tricky to ensure that the "callback" is always despatched to the correct function.

Let's Move Outside APL

```

)load isidde
saved 1993-05-21 09:47:18
wd'winexec apl.exe'      * start an new apl session

```

"DDE is a very badly designed and terribly implemented version of shared variables — but everybody uses it!"

Top window ... let's be a server:

```

wd'ddename serv;'
□ ← d ← '*data' wdg wd 'wait;'

```

The *wdg* is a simple utility to extract the **data* item from the result of the *wait* command.

Bottom window (top window is waiting):

```

wd'ddepoke serv topic item "how now cow";'

```

... top application displays "how now cow" and returns to prompt. Of course where DDE really pays off is when communicating with something other than APL — but learn it with *APL←→APL* so you are in control of both ends of the link.

Q: Aside from getting the link started, is it really asymmetrical?

A: No — they've done a lot of work to make it asymmetrical, but you can overcome that!

```

runexcel ... start Excel at 'sheet1'
2 2 set ?3 5p100
saveas 'mun.xls'

```

The numbers show up in Excel, and get saved as a spreadsheet. You can fire quite complex command sequences from APL, but it is usually better to work up

the macros in the Excel environment (where you have access to the Excel debugging tools) and simply fire the whole macro from APL.

This also works with MS Access ... but someone has been fiddling with the demo! Come and see it on the stand — but it does work, honest!

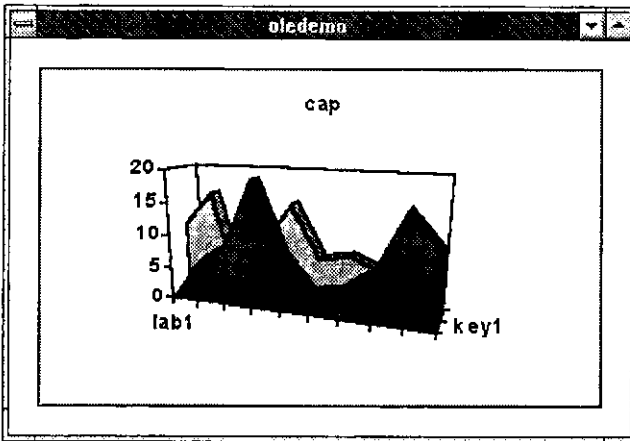
Object Linking and Embedding

Read the Microsoft books, and try to figure out what this is ... or just use it!

```

)load isiole
saved 1993-05-21 09:47:50
demomsggraph a runs MSGraph as an OLE client.

```



```

graphdata 10p2
graphdata ?2 10p20 ... etc

```

If you have an Excel user who is a hot-shot at graphics, he will instantly know how to use your application!

Other useful OLE servers which will sit inside your applications:

- Paintbrush (everyone gets this with Windows, and most people can use it).
- Object Packager — this makes an excellent 'launch-pad' for other applications. They can be packaged up with an icon so they sit happily on a button in your system.
- most modern Windows applications (from Excel to CorelDRAW!) will act as OLE servers.

Summary

You don't have to be a masochistic C programmer to be a GUI programmer! The window-driver will shortly be packaged up as a DLL — it is probably easier to do it than to do the market survey, so we'll just do it.

Q: Motif?

A: Right now it's very hardwired to Windows. We believe it will trivially port to OS/2 PM and it should be reasonably easy to move to Motif.

The session closed to general acclaim.

Commentary

by Adrian Smith

One of the most useful innovations of this conference was to repeat all the tutorials during the main body of the programme. This gave the participants a chance to do some serious learning, and made an excellent opportunity for us to break the normal routine. I found Eric Iverson's 3-hour session on the ISI Windows Driver most illuminating; Eric is an excellent teacher, and introduced the material in a logical and 'easy to swallow' sequence. He also fielded questions thoughtfully, and was always willing to repeat material (using different words) for the benefit of the non-English speakers in the class.

Personally, I find myself wanting to code in the union (not unfortunately the intersect) of Eric's GUI and Dyadic's. I think they both have things to learn from each other, and if they can move just a little way to closing the gap, I suspect that between them they have the scope to clean up on the world market for PC APLs over the next 10 years.

In no particular order:

- I understand Dyadic's reluctance to provide native access to all the Windows common dialogs (these don't exist under Motif), but the application hack in me wants a quick way of getting to `Fontbox` and `Colorpalette`. Maybe a distributed `WS` to cover `□NA` would suffice?
- Eric has rightly shied away from reporting all the mouse-move events, but these are pretty useless anyway. What we do need are 'mouse-up' and 'mouse-down' to allow us to put up pop-up menus at the mouse location. More useful than anything in `Dyalog` would be a 'mouse-entered-object' and 'mouse-exited-object' pair — I don't know if these could be done, but I know I would like to use them!

- Dyalog are going to have to look into OLE. The stuff that Eric showed with MSGraph and Paintbrush was clearly the way forward for simple business applications. If I want a table in Word, I simply embed a spreadsheet — far faster and much more reliable than the Word 'tables'. If I want a simple chart on a form (which my user can edit) I should use MSGraph.
- I really like Dyalog's 'locator' object. Doing your own 'rubberbanding' with APL needs a snappy 486 — the locator does it for you and gives you an event when you let go the mouse. Eric, please copy!!

The final comment I would like to make is a tricky one about the whole philosophy of programming with callbacks. The old-style programmer in me is much more comfortable with Eric's approach of returning to APL on every event and letting a bunch of GOTOs handle the event logic. This way I can see what is going on, and I can execute arbitrary slugs of APL code when events happen.

On the other hand, I have to say that (so far) the systems I have written with Dyalog have been quite astonishingly reliable; far more so than I have ever achieved in the same time with a conventionally structured workspace. I also managed to code Minesweeper with precisely one branch arrow, and that was only needed due to sloppy thinking.

I think that what I am asking for is:

- a simple table-driven event handler to field the `⊞wd` returns and conditionally execute the corresponding code. Wildcards or inheritance could be explored. This could also include a simple 'monitor' to help me to see what was happening. Basically this would be doing the same job as Eric's *напма in* function, but would be a lot more extensible, and much easier to debug.
- something in the Dyalog 'trace' option to let me follow what is going on during a dequeue. Because I never get back into APL until my program ends, the conventional tracing methods are almost useless. If I fire a mis-spelled callback, all I get is a `VALUE ERROR` with no clue where to look for the mistake! This is all quite awful, and has driven me to the brink of rejecting Dyadic's approach altogether and reverting to something closer to the ISI philosophy.

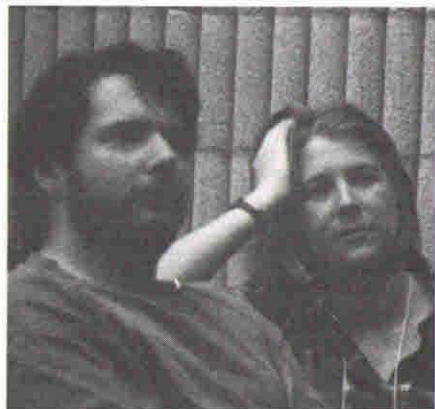
The basic problem is that neither Dyalog nor ISI are programming applications with this stuff! The rest of us are just beginning to climb the learning curve, and don't have the expertise to offer them any really sound advice. Nevertheless, I think we should try, and I would also advise owners of either interpreter to go and try out the other one! That way we stand a chance of getting them to converge, at least to the point where a common utility set could make the differences irrelevant.

OPENING PLENARY

Larry Moore 'Welcome to Toronto'

reported by Anthony Camacho

Larry began by thanking many of the people who had made APL 93 possible: Lee Dickey who has produced an excellent programme, Kirk Iverson who has put together a software library of 30 Mbytes, Peter Wooster whose APL Art Gallery is on show upstairs, Vin Grannell who bullied everyone until they arranged to record the presentations on Video tape, Elena Anzalone who edited the Proceedings (and also produced the Program and other conference documents), Bruce Bolin who had managed the exhibition, Ben Best the Treasurer whose efforts (and Bob Bernecky's) had raised the funds to enable the Russians to be invited, Marc Griffiths who had handled publicity, Cameron Linton the APL 93 secretary without whose minutes recording actions and policies much less would have got done, and not least Bob Bernecky who (in his role as Lord High Everything Else) had done everything that other people hadn't got around to.



Kirk Iverson and Elena Anzalone



Larry Moore

He spoke of the conditions which were necessary to run a conference. Few APL Sigs could do it. To form one you need a group of people, a location to meet, an agenda, speakers, advertising, a mailing list, a newsletter, a bank account and a

mailing address. The Toronto group was formed in 1981 by Dan King. It now has 50 members and 300 on the mailing list and holds monthly meetings from September till May. He praised Cameron Linton, ex-secretary and now President and Jennifer Kieffer the last President. Ed Shaw complained about the rather high proposed charge for the delegate list on disk: the price was subsequently reduced.

Donald McIntyre 'The Triumph of Symbols over Words'

reported by Anthony Camacho

Gene McDonnell introduced the speaker: he told us that the conference organisers had learned, too late, that a pibroch had been written entitled *Donald McIntyre, Kinfauns*: they would have loved to be able to play it to us. Donald then began and about two sentences later there was a loud wail and a Scottish piper in full regalia entered at the back of the hall and marched onto the platform playing the pibroch.

The two sentences, that Donald spoke before the pibroch began, caused great anxiety among the organisers: they had not realised that the pipes have to be pumped up before starting to play so the cue should have been given 30 seconds earlier. As Ken said 'It reminded us that we should not make assumptions about technologies we don't understand.' But I don't think Donald minded the interruption. *Donald McIntyre, Kinfauns* appears in *The Piper* for September 1993.



**Jimmy Connolly plays
*Donald McIntyre, Kinfauns***

SIGAPL Annual General Meeting

reported by Jonathan Barman

As usual, SIGAPL held their AGM at the conference. This is an excellent idea as it enables a large number of members to attend. The chairman, **Dick Bowman**, introduced the new SIGAPL board and gave us a brief view of the objectives of the board for their term of office. They plan to focus on improving the service to members, which includes getting Quote Quad out on time. However there is a considerable lead time on making improvements and Dick felt that members would start to see significant results in about 12 to 18 months time.

Stuart Yarus is vice-Chairman. **Mike Kent** is in charge of finance, and reported that they have \$2,500 available and expect \$10,000 from St Petersburg. For the level of business carried out by SIGAPL they would ideally have reserves of around \$50,000, so there is quite a bit of work to be done to improve the finances. There are about 700 regular members, which includes ACM and SIGAPL, plus 150 SIGAPL only members, and 200 who just subscribe to Quote Quad.

Ray Polivka is editor of Quote Quad. His main goal is to achieve a consistent and timely delivery of the journal. The June 93 issue is currently with ACM for delivery, so it should not be too late. Ray plans to set up working groups to help with all the tasks that are required to get Quote Quad out on time and to make sure that it has a good supply of articles for publication. It is also hoped to publish refereed papers in *Computing Reviews*, for which reviewers will be needed.

Robert Brown is the conference co-ordinator and is evaluating the alternatives for the APL95 and APL96 conferences. Other members of the board are Lee Dickey, David Weintraub, Curtis Jones, Marc Griffiths and Lynne Shaw, the ex-chair. Dick Bowman wrapped up the session with a plea for local APL groups to become chapters of SIGAPL.

A member of the audience asked what were the advantages of becoming a local chapter of SIGAPL. Dick Bowman said he would talk to the questioner later! However, there were advantages in the publicity available in Quote Quad and the involvement in conferences such as Tool of Thought. Jacob Brickman, chairman of the New York SIG, said that the Tool of Thought proceedings for 93 were now available and also back issues. Dick Holt said that TOMAC got increased professional help by being a local chapter.



Ken Iverson "On the Steps"



Sasha Skomorokhov and Dmitri
Lukyanov with a present for Ben Best



Marc Griffiths

PRESENTATIONS

Pierre Deslauriers 'APL Helps the Deaf to Hear Again'

reported by Jonathan Barman

When a person becomes completely deaf there are still things that can be done to help. It is possible to implant a device by the cochlea which can stimulate the aural nerves. To avoid having a break in the skin the device is completely self-contained and is enclosed in platinum and sapphire and it has eight small probes which make connections with the nerves.

Outside the skin a transmitter sends both power and signals to the the implant. The sound source is a microphone and its signal is processed by a sixteen bit fast digital signal processor into the signals to pass to the implant. This processor can handle 15,000 measurements a second as well as respond in real time to control signals from a computer. Its output is up to a million bits a second and this it converts into suitable signals to send to the implant.

The digital signal processor has 250 control parameters and these may change the signal to the implant through a much wider range of values than the patient would find comfortable. Since the eight signals available are at best a poor replacement for the original signals on the aural nerves, and since the surgeon cannot be sure exactly which nerves will be stimulated by each electrode after the implant has healed, there is a big problem in setting the signal processor controls.

This problem is solved by a control setting program in APL. This allows various representations of the signal and controls on a colour screen to be adjusted by the scientists, doctors or patient while the patient can report the effects on what she perceives. The system allows limits to be set on each electrode separately so that settings which get near to being painful are excluded. Then uncomfortable combinations can be excluded and finally the patient can experiment to find settings which give stimulations which she feels she may be able to learn how to interpret.

The patient in these first experiments was made deaf by treatment for hepatitis B, and it is sad to report that she has since died. The ability to adjust the signal processor and to have every aspect of the signals adjustable and under control, was felt to account for the comparative performance improvement over previous cochlear implants whose success has been limited. The patient said the sound was very much like 'Mickey Mouse' (Nature is hard to replace!) but the aid it gave to lip-reading was immense. M. Deslauriers wrote: *Watching the patient's expression when we first started the DSP and said "Hello" produced an unforgettable feeling that could not be matched by anything.*

Tribute was paid to Jacques Beaulieu PhD FRSC (Fellow of the Royal Society of Canada) who developed the neurological model from which the design was developed for the way the cochlea was stimulated and the clinical tests. The conclusion was that APL is an outstanding tool for research and development.

Robert Bernecky 'The Role of APL and J in High Performance Computing'

reported by Anthony Camacho

Bob began by saying that the paper is in the book and he proposed to illuminate some of the problems and show how APL is a superior tool for addressing them.

He displayed tables showing degrees of parallelism, connectivity and program characteristics:

Degree of parallelism	Number of processes	Connexion	Machine
Serial	1	Bus	PC
Dinkily parallel	2-16	Bus	S/370
Moderately parallel	16-1000	Hypercube	NCUBE CM-5
Massively parallel	over 1000	Mesh cube	CM-2
Connectivity			
Bus	Minimum circuitry; low bandwidth; not scalable		
X-bar	High bandwidth; prohibitive circuit count		
Hypercube	Scalable but collision delays		
Mesh	High bandwidth for near neighbours slow for long distance		

Program characteristics	Type	Decomposition Method
Level of parallelism		
Low	Coarse	Few large pieces
Moderate	Coarse	Few large piece
Massive	Fine	Data Parallel

He drew crude graphs of computing time against the number of elements in the arguments. These showed a fixed start-up time and then a gradual increase as the number of elements increases. More sophisticated algorithms increase the start-up time but reduce the slope of the graph; when the average number of elements is large enough it pays to use a more sophisticated algorithm. Compilation greatly reduces the start-up time and also reduces the slope of the graph but of course requires a preprocessing operation.



**Stephen Jaffe and Bob Bernecky
at the Closing Picnic**

The CM-2 machine is a single machine handling multiple data. SISAL is a Single Assignment Array Language, which easily beats FORTRAN on super-computing machines. Bob Bernecky aims to compile APL into SISAL.

A graph showing compute time per element is a curve which approaches a minimum at an ever-decreasing rate. This can show two key values, the number of elements at which half maximum speed is reached and the (arbitrary) level at which the speed cannot for practical purposes be increased.

Bob Bernecky's aim is to be able to use 90% of the capacity of a supercomputer such as CM-2 on a wide range of programs. Whereas supercomputers, which are very expensive boxes, used in practice to be special purpose machines, each bought to run a single special program, the future is for general purpose supercomputers which can be used to run many different programs.

Gérard Langlet

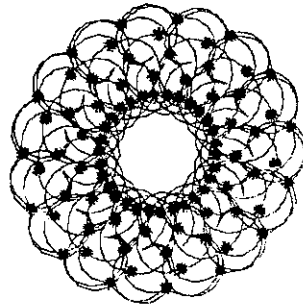
'Building the APL Atlas of Natural Shapes'

reported by Anthony Camacho

Gérard says the world is driven by symmetry and that information should also be driven by symmetry. He had asked some physicists to describe a man by a system of equations. They thought the task impossible, yet simultaneously the human genome project is gradually identifying the equivalent — a specification of a man using a representation with only four states per element.

The fundamental nature of the function not-equal scan (which he conjectures may be the general problem solver sought by so many people for so many years) led Gérard to ask himself whether it could be used to generate shapes; not just simple geometric shapes but the shapes found in nature. One of the things he had already found was how the function generated the Sierpinski gasket.

The experiments were rewarding. Beginning with the Koch snowflake curve (to establish a procedure for converting arguments into graphics), we were then shown a large number of the outputs from the function SHAPES. Apparently the not-equals scan function is the equivalent, for binary sequences to Riemann-Liouville integrations and differentiations for continuous functions. There were other fascinating analogies drawn, for example between Gérard's cognitive and helical transforms for bit streams and the Fast Fourier Transform for continuous functions.



Gérard then showed us a large number of pretty, mainly circular, patterns which his functions had drawn. The workspace for APL*PLUS/PC is in the software library. It was written in ISO APL (with a single extension — replicate — which is now implemented in all APLs, even I-APL) so it can be adapted for any current interpreter by writing the implementation-dependent plotting routines. The version in the library promises colour displays.

Workshop — A GUI Standard for APL — Facilitator: David Crossley

reported by Anthony Camacho and Jonathan Barman

Although billed as a workshop, this was really a panel session with the usual introductory statements followed by audience comment and questions. We had not realised that there were two David Crossleys in the APL world, one who lives in Canada and one who lives in France. The Canadian David Crossley introduced the speakers: James Wheeler, David Liebttag, Peter Donnelly and Eric Iverson. He explained that he had written two systems in APL*PLUS II using quad-WIN and that he doesn't want to learn many windows GUI standards. He supposes the rest of us are equally reluctant and so we need to agree a GUI standard.

Initial Statements

James Wheeler, who has been with STSC/Manugistics for 16 years, told us that Manugistics were committed at least to emulate the behaviour of the current features of their interpreters, so users could write new systems in confidence that they would continue to run in upgrades. He was dissatisfied with the current GUI interface and felt that we are far from reaching a final solution. It would be inappropriate to set a standard until we are sure it will last.

Peter Donnelly, who has been with Dyadic for 14 years and has worked on GUIs for the last three of them, tended to agree that it was too early to standardise. He felt that the GUI interface eventually needed to be part of the language, and to be portable would have to work not only with Windows, Motif and Presentation Manager but with any GUI system. A standard would have to apply to all GUIs. At present competition and the desire to have a competitive edge are strong forces against standardisation. In the process of looking at many GUI toolsets Peter has found that *Objects*



Peter Donnelly

and *Events* are easy in APL but that *Properties* are unnatural to it and the Dyadic team had implemented them as quad-functions. The *Methods* used in Visual BASIC for specifying circle, rectangle and so on are also unnatural in APL. Peter thinks that Visual BASIC has got it wrong. Dyadic found that call-back functions are easy and natural in APL functional structure and so overall APL is a great way to do GUI programming.



David Liebttag

David Liebttag had ten years of applications experience before his work with the APL group for the last three or four years; he has recently been working on the OS/2 user interface. He said he admires the Dyadic approach but thinks it is wrong, and so he agrees that it is too early to standardise. Simple GUI interfaces are easy but a full one may be very complex indeed. As in consequence it will be expensive, it is not worth doing unless it is going to last; a two year life would not pay back the investment. In David's opinion the functionality should be in the operating system, not in the language, so IBM's current emphasis was to provide lots of easy-to-use utilities.

Eric Iverson has been an APL implementer for twenty-five years and has been working on the GUI interface for two years. He spoke about the philosophy behind the ISI window driver. The primary design goal was to keep the interface at a high level so that it is easy to use (N.B. NOT like C). A second goal is to make the driver portable to Motif and other windowing systems. So ISI began with the essentials to cover the needs of standard applications omitting the less often needed facilities but allowing the design to be extended when ISI users needed more. Using a windows driver meant that the same mechanism could be used for both J and APLWIN and so the commands to it are written in a scripting language like PostScript or SQL. This approach preserves the integrity of APL and also makes it possible to sell the driver separately so that it could be driven from any other APL or any other language. ISI intends to improve the windows driver by adding features, but not too many low level controls as ISI does not want to compete with Visual BASIC or Smalltalk. Eric hoped the DLL as an independent product would be put on sale.

Contributions from the Audience and Panel Responses

Ray Polivka confessed he was floundering; he had not been given a clear idea what it is that is involved in standardisation and still wanted to know.

Eke van Batenburg of the University of Leiden wants to be able to port GUI workspaces between PCs, Macs and Ataris. APL porting is easy; GUI porting is hard. He suggested that if we wait too long before standardising we will get another bad split, like that between boxed and nested arrays.

Jim Ryan said that he had been using APL for 25 years and that he would be satisfied if APL communicated with the GUI in a standard way. His preference is for the Dyadic solution to the problem. Willi Hahn later said that he would like to see the equivalent of quad and quote-quad for GUIs (whatever that would be). Steve Hayward asked about the value building GUI interfaces into APL when there are lots of products available to do the same job.

Peter Donnelly replied that Dyalog wished to be able to compete with Visual BASIC, C applications and that using another product to drive the GUI would be giving up the competition.

Adrian Smith said he sat alongside a user of Power Builder and could do the same job in a fifth of the time because he was using APL. He liked the scripting approach — his system writes PostScript in APL — but the problem with scripting is when you want to control the interaction between the user and the system.

Eric Iverson said that ISI had solved this difficulty; a scripting language can handle call-back and so this doesn't have to be a disadvantage of scripting.

Gregg Taylor suggested that he could map most ISI features to Dyalog features and vice versa; he felt these two implementations were not far from as compatible as two implementations of the APL filing system used to be. On the other hand he had found he could not do the same thing with IBM or Manugistics interpreters.

James Wheeler said he was 100% in agreement with John Scholes (of Dyadic) about the conceptual framework of *call-back*, *objects*, *properties* and *methods*. The differences between Dyalog and APL*PLUS arise from the things that are not intrinsic to the language: he pointed out that features built into quad-functions had not been added to the language, only to the implementation.

Richard Eller of TMT-Team said that using APL with an independent GUI product adds a lot to the complexity.

Asked why, as everyone agreed that APL was good at *objects*, they could not be added to the language, Peter Donnelly explained that the difficulty is that APL does not have *structures*: a *structure*, for example, would be a defined pattern for a nested array. David Liebttag amplified by saying that a *structure* would be the equivalent of a user-defined type.

A speaker who claimed to come from Harvard (but no-one is listed as such in the list of delegates) suggested that even if we could agree on a specification for the ideal GUI interface, the thing would necessarily be so complex that there would not be enough users to test out all its features and find the bugs in anything like a reasonable time.

Richard Levine thought a good approach would be to emphasise the importance of examples. Try many out in several implementations and compare the elegance and ease of use of each of them. By publishing such comparisons Vector and Quote-quad could serve the community by helping people towards an informed consensus.

Arto Juvonen said he had a *Vitaform* driver which could drive Dyalog and Manugistics with a scripting language. He had a window looking like an HP45 calculator containing 65 objects which worked in both implementations.

Nikolai Puntikov said that maybe it is too early to standardise the GUI interface but it is most important to be able to do the work using the currently available operating systems and interpreters. To get what he needs he writes DLL code in C and so wants a way to drive this code. It is how APL should control this DLL driver that ought to be standardised. Since an object oriented APL would be a different language, there is no point in trying to incorporate it inside APL.

Zdenek Jizba said that he had been using APL since it was invented and that Iverson's genius was to abstract the concepts necessary to express algorithms of great power. He felt what we needed was a similar genius to abstract the essence of the GUI interface.

Adrian Smith said that no-one has yet had to maintain an APL/GUI system. He foresaw problems and wanted to agree with Nikolai that putting the GUI features into APL, even as quad-functions, compounded the difficulty.

Eric Iverson said that the vendors have provided enough to enable users to write APL GUI systems and that by sticking to the middle of the road users could write fairly portable GUI systems now.

Richard Levine wondered whether the APL Toolkit could be expanded with suitable GUI cover functions.

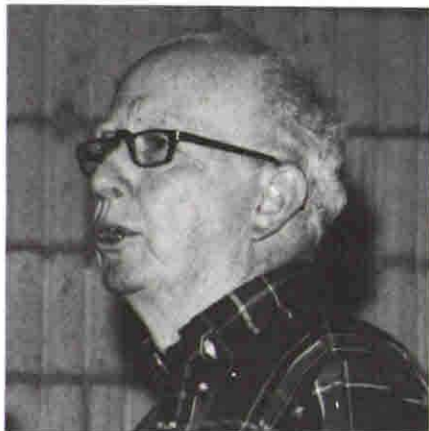
Keith Sanders wondered whether APLs could be given access to the DLLs in Visual BASIC, but James Wheeler said that Microsoft forbade the use of the necessary hooks.

Eric Iverson hoped that the DDE engine used in APLIWIN and J-WIN would be available in the autumn.

David Steinbrook and Eugene McDonnell 'From Trees into Boxes'

reported by Anthony Camacho

David Steinbrook introduced the presentation by saying it was a report on work in progress. The authors had been given access to the source code for J from the beginning of the project. The original proposal had been to add a tree *data* type to J, encouraged by Knuth's remark that trees are *the most important non-linear structures arising in computer algorithms*. The version of J with the new data type was written but there were many overlaps between the operations required to manipulate trees and those provided for boxed arrays.



Eugene McDonnell

On investigation they found that the J boxed data type enabled them to do even more than they had expected. All they needed to implement trees were some specialised verbs, adverbs and conjunctions and so they now have implemented trees as a subset of boxed arrays. This has been done without adding a new data type to the language. The new parts of speech are added as external verbs in the 12!:n range and the C code is very similar to the J model on which it is based.

The main body of the presentation followed the paper fairly closely, going through the representations of a tree,

the process of tracing a tree and over fifty facilities for manipulating trees or parts of them. The traditional representation of a tree is of the trunk and its branches; there are no roots (which would make it look double-ended) although the node where they would go is called the root, yet for some reason a tree is usually shown as Ken would put it *upside down* with the leaves at the bottom. We were told that Ken also says trees are round (3D), although the data type consists entirely of espaliers (2D): how this affects things and how the paper representation falls short was not made clear.

David said that the source code would be made available and looked forward to seeing trees in common use in J.

Robert Bernecky 'Array Morphology'

reported by Anthony Camacho

Robert Bernecky began by saying that the reputation APL has for being slow is deserved when it is used to do lots of operations on small arrays. This is because to execute a line of APL it is necessary to perform a series of initial tasks:

- Analyse the syntax and put together the actions to be performed
- Decide the storage needed and allocate it
- Check the type of each variable and intermediate value for compatibility
- Check the form of each argument to ensure they conform as necessary

These tasks vary little with the size of the variables; for a line which processes scalars the overhead is almost as much as for a line handling large arrays. When the large array has to be handled by a parallel processor there is extra overhead involved in deciding which bits to send to which processor and this adds to the overhead. According to Bob the average number of elements in an array is only about seven to fifteen. This is far too few for efficiency and the initial tasks require more work than the processing of the variables. It is therefore important to write APL so as to use to the full its ability to process large arrays and this becomes even more important when one wants to use a parallel processor efficiently.

To use a parallel processor in APL one has either to change the language by adding declarations of variable sizes and types or, if one is unwilling to change the language to be able to deduce these. Without changing the language it is possible to add lines of code which force a variable to take a particular type and if, for example they passing a localised variable which is not used the analyser can also omit the processing implied in the line from the final compiled code. The deduction can be done on a preprocessing pass through the code. It is surprising how much can be deduced from perfectly ordinary code; for example in Mike Jenkins' model of domino the Bernecky analyser was able to deduce 100 out of 175 types. Adding just a few declarations raised the total to 174.

The C code produced by the analysis and compilation is often not nice. No-one would want to maintain it. It may then be optimised and after that it may be unrecognisable.

In general the improvements that J makes to the APL syntax reduce the deductions that can be made. For example the APL expression $X[1;]$ enables the analyser to deduce that X has rank 2 but the J equivalent does not give the equivalent information. However declarations by additional code which the analyser can recognise as omissible are equally easy in J and some features of J enhance compilability.

Norman Thomson 'Understanding ANOVA the APL Way'

reported by Sylvia Camacho

Norman is an expert and charming teacher and he began in characteristic style by saying that the consequence of devising the functions he was about to present was that he now feels that he understands ANOVA fully for the first time. The implication was also that the subject is a good deal simpler and more approachable than it has been made by professional teachers of statistics. Indeed this was the theme he was developing when he suddenly broke off to present his last slide in order to ensure that he would not run out of time before presenting his conclusions. This seemed to be an exercise in canny Scottish prescience because two minutes later the lecture was prematurely ended by a false fire alarm.

The paper contains all the functions, of course, beginning with the basic set,

- Mean of a vector
- Vector with mean substitution
- Sum of squares about the mean
- Mean polish of array — all axes
- Fitted values following regression

These are then used to show how to perform:

- One-way ANOVA
- Two-way ANOVA
- Multi-way ANOVA
- Analysis of Variance following Regression

The use of the regression function is then developed in a discussion of orthogonal polynomials, regression with two-way data, multi-way ANOVA with regression and maximum decomposition of sums of squares. Finally an illustration is given of the application of the functions to experimental data.

To round it all off Norman provides a Summary of the Principal Functions,

- one-way ANOVA
- two-way ANOVA
- multi-way (main effects)
- multi-way (all effects)
- multi-way interactions
- 1-way with linear regression
- 1-way with polynomial regression
- 2-way with polynomial regression
- multi-way polynomial regression
- 2-way total decomposition
- Balanced incomplete block designs
- Latin & Youden squares

For maximum impact the following message should be projected as large as possible on the wall at stage back and the lecturer should leave the podium without further comment, allowing the audience to draw its own conclusions.

‘These functions have been fully described and their use illustrated in less than ten pages. They are fully sufficient to deal with the computational needs of the many and varied exercises in one of the best known current textbooks “Design and Analysis of Experiments”, Douglas C. Montgomery, 3rd edn., John Wiley & Sons, 1991’

Yet we wonder why APL is not popular with the Educational Establishment?

Jack Rudd 'APL in Satellite Surveillance'

reported by Adrian Smith

Now it can be Told!

Much of what follows has been hinted at over the years in a series of papers (e.g. "An APL to ADA Translator" in 1987), but Jack has never been able to say exactly what he was doing, and why. Now that the "cold war" has relaxed, and the threat from Soviet ICBMs has almost totally vanished, much of what Jack is working on has been de-classified. He was clearly delighted at being able to talk freely about his work with APL in the analysis of satellite observations, and was also willing to offer some strong opinions (see below) on the future.

All in all, I felt that this was one of the most significant papers at the conference, and judging by the attentiveness of the audience, many others would agree with me.



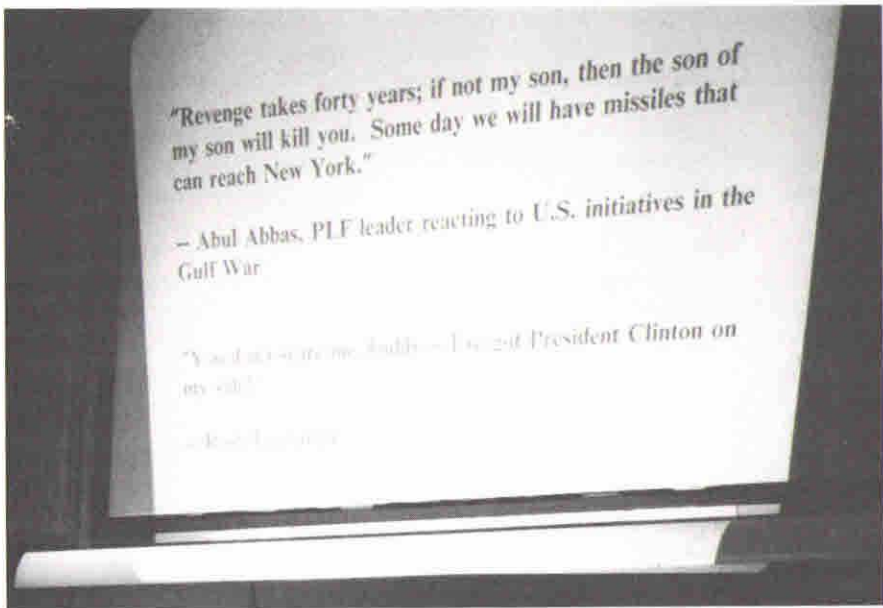
Don Farrand, Roger Moore, Jack Rudd, Denny Jizba, Vin Grannell

An Announcement

"Ballistic missile defense is now a possibility:

- the threat is reducing
- the \$/performance ratio of sensors is improving
- we now have far greater computational capability
- we also have 22 years experience of algorithm development"

The experience of defending against SCUD attacks in the Gulf War, when satellite observations were relayed within (some classified piece of time) to ground stations, and the use of PATRIOT batteries was optimised in "near real time" confirms this.



Where is APL in All This?

- **data analysis.** Lots of satellites have historically seen lots of launches! There is now plenty of scope for verifying algorithms against past data.
- **algorithm prototyping.** APL is used to benchmark code (via quadNA to FORTRAN), to visualise algorithms, and often (via indirect translation to ADA) to specify the final software. "English is a very inefficient intermediate language, so we give them APL".
- as of last year — APL is being delivered direct to satellite analysts.

History

Things started back in 1971 with an IBM/360 (1+2Mb) which offered a 24hr batch turnaround — unacceptable when Soviet submarines were just offshore! APL over 2741 terminals was a huge improvement, even when a bad sunspot year gave so many RESENDS that it was hard to get a clean line of code. In those days, the threat was very real, and Jack's group consistently met aggressive schedules. Today, it takes forever to get anything done — “you do not want to be a Type-A personality and work with ADA!”.

In 1985, Jack developed a translator from APL to “readable ADA”; the developers said they would use it, but by 1988 the ADA folk had gone OOPS — and their defined types pervaded the ADA environment. All APL wanted was INTs and LONGs and FLOATs — so the translator was not used; instead Jack hand-translated the generated ADA to FORTRAN as you cannot call ADA code with quadNA (ADA was designed to be the only language in the world). The ADA developers eventually re-translated the FORTRAN source into ADA anyway (by hand). However, although the route seems to have been a little perverse, the consequence is that much of the real-time ADA code that is used to defend against SCUD attacks closely embodies original APL algorithms.

Satellite Constellation Management

The requirement is quickly to get satellites in the right place to view a specific piece of ground. For the first time, raw APL2 code (around 1500 lines to date) is being delivered direct to the analysts. The encouraging thing is that the analysts have already picked up enough APL to be able to operate and modify the code themselves. However:

“Our analysts have no complaint with your program. I want state-of-the-art displays to impress passing generals.”

... is the message from the management! Accordingly, Jack has organised an APL2 course in October, with the objective of getting the interface GUIfied.

A Catalogue of Space Objects

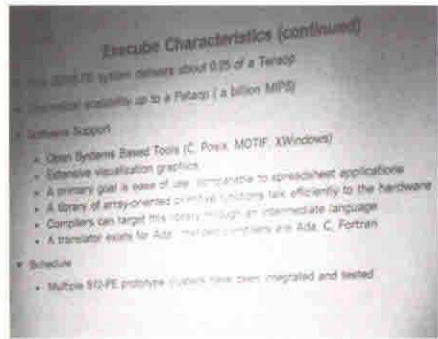
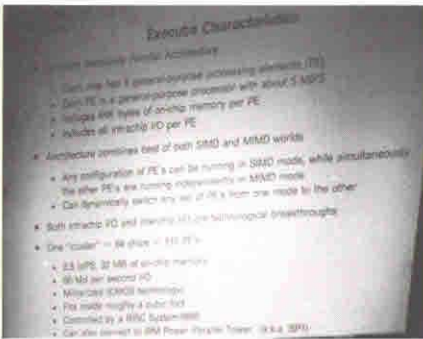
Some 7000 objects are tracked -

“Things that if they hit you, going at 18,000mph, they're going to disturb your day!”

... but to track them all accurately would require around 1Gflop of processing power. However there is hope, because this is almost a perfect parallel

processing application. The process starts with a detailed model of the Earth that includes every gravitational bulge. Lots of fancy numerical integration leads to well parallelised orbit-propagation algorithms which are ideal for the IBM "Power Parallel Tower" processor. The objective is drastically to reduce the need for intervention when the software tries to guess which observation $\leftarrow\rightarrow$ which object (at an average rate of 50,000 per day).

The "Parallel Tower" is based around the "Execube" processor, which is designed to be a scalable, affordable, massively parallel architecture. In its basic form, it will process 2.5 billion instructions per second in about 1 cubic foot. What is more, it is already "militarized"; it was designed to travel in AWACS aircraft, and could easily be put in a satellite. Many array-oriented primitives are wired into the processor (which was developed by a team with APL experience); the irony is that it requires a translator for ADA!



Questions

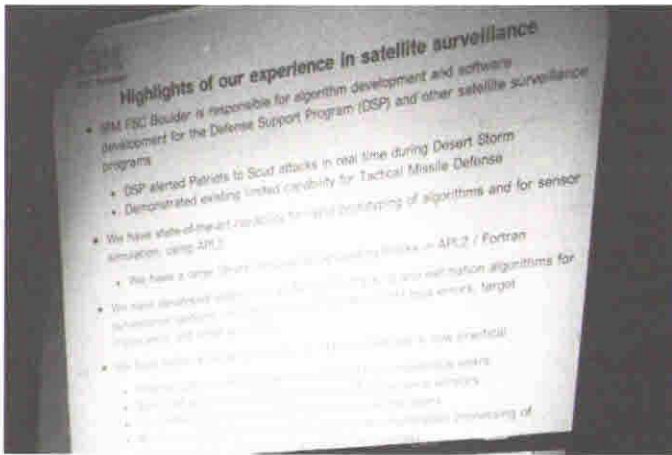
Q: Do you need to track geostationary objects?

A: Actually, there is no such thing as a geostationary satellite. Anything left to itself will build up an oscillation of around 4 deg in a year, unless you expend energy in keeping it in place.

Q: Do you attempt "provability"?

A: Provability is something we've never been able to afford. Only 100 million people are involved if we get it wrong, so we can't afford the \$1,000 per line of code that the Space-Shuttle people use!

A: No, I don't know anything about UFOs!!



Commentary (by Adrian Smith)

Just as C hung around from 1976 until the mid 1980s, waiting for the right conditions to flourish, APL has been there in the wings quietly hoping that eventually a suitable computer might be built. Jack's talk gives us hope on two counts:

1. The hardware has almost caught up! APL was never designed for a von Neumann machine, and soon it will no longer be constrained to run on one. I am not aware of any other high-level languages which are "parallel-ready"; ADA obviously isn't.
2. The CASE paradigm (design, verify, code, test, implement) seems at last to be running into the sand. When the pressure is on, the need to get results rapidly overrides the desire to operate "by the book". Languages which build CASE deeply into their architecture (as ADA does) will die with it. Functional languages (like APL, LISP and PostScript) are best placed to exploit their downfall.

To repeat what I said at the start: this was a significant paper in the history of APL.

Birds of a Feather Session on Newsletters

reported by Anthony Camacho

Those present were: Stuart Yarus, Diane Whitehouse, Rosemary Snow, Jonathan Barman, Anthony and Sylvia Camacho, Marc Griffiths, Cameron Linton, Jennifer Kieffer, Ray Polivka, Curtis Jones, Dick Holt, Jacob Brickman, Alan Mayer, Zdenek Jizba and Gregg Taylor. As everyone has a different first name, surnames are omitted below.

Jacob said there was a problem when editor and publisher were the same person. This was often too demanding for one person and even if the first editor sustained it the second often could not.



Editors of Education Vector and Vector with David Eastwood

Anthony described the arrangements for Vector: a large (about a dozen people) and friendly working group who all could and did solicit as well as write articles so there was not too much work for any one of them, a production manager who did final layout and typesetting (and who was paid, though not perhaps at a commercial rate), an administrative assistant (also paid) who chased advertisements and sustaining member news and who kept the product guide up to date. Of course there is also a paid printer and a paid mailing house. There is an income from advertisements which enables Vector to pay reporters

occasionally and to cover the expenses of all concerned, with the overall result that the editor could even do some writing in addition to the editing.

Curtis liked the diversity and the rapid response of the small newsletters.

Ray wanted more readers and to meet the needs of the readership; he thought Quote Quad should reprint some of the excellent material from the SWAPL newsletter (recently renamed the APL Perspective), which would otherwise only reach a tiny audience.

In a comparison of readerships the Chicago newsletter reaches 50-100, Education Vector 1200, Vector 900, Quote Quad 800, the New York newsletter 300, the Toronto newsletter 250, the Bay Area User Group newsletter 75, the APL Perspective 200, the southern California newsletter 40.

Gregg wanted to get back to the problem of burn-out, which was a risk arising from success. When he edited SWAPL newsletter he wrote it, edited it, typeset it and published it. When he began it was four pages of quarto and it grew to ten. Now the average is 20-24 and the copy included in the delegate pack is 36 pages.

There was a discussion on reprinting material from other newsletters. Ray said Quote Quad is supposed to be original material but he would like to republish material which had only a small circulation or to publish things jointly. Alan pointed out the copyright problem. Vector leaves the ownership of the copyright with the author and the author assents to the permission given inside the cover to anyone to reprint with acknowledgement; Quote Quad requires the copyright to be assigned to it and Vector (among others) cannot do this.

Rosemary pointed out that there were many newsletters or magazines not represented at the meeting — the French, German, Japanese, Finnish and Russians to name but a few.

Jacob said that de Kerf had reprinted a great deal and used not to give the attributions 100% of the time but that now he does.

Curtis said that it is always worth asking before republishing as frequently there is an improved version (even if all that is changed is the typos).

Jennifer pointed out that the purpose of a local newsletter is quite different from an international magazine such as Vector or Quote Quad; it has to announce meetings for example.

Gregg wanted newsletters to see themselves as propaganda vehicles. Every issue should bear an uplifting message and he would like to see this message

appearing at least monthly. Jacob thought the people who produced the SWAPL newsletter should be commended and should tell the rest of us how they managed it.

Diane said that the key was to get a team together to share the work. If the team set themes and solicited articles, gave themselves an aim and saw the main steps to achieving it then success might be possible without burn-out. Jonathan agreed that the main problem was getting enough articles.

Jennifer suggested that everyone present commit themselves to sending a regular report of what is going on to all other newsletters.

The meeting adjourned to the cafeteria at 12:15, but the reassembly was not wholly successful; after a short while eating we moved back to room 3163.

Zdenek suggested that a computer club could circulate material such as workspaces on disk. Many people would contribute workspaces when they might be reluctant to write them up.

Dick asked everyone to send reports to him and he would put them on his bulletin board.

Curtis suggested that we each send all the rest copies of our publications: we can do this now with very little effort. Ray suggested that we include membership details and ask each to join the other groups.

Ray was challenged, on his joint publishing suggestion, to list what information should be shared and reported to everyone. Jennifer said that the Toronto group usually know two or three months ahead what their activities are going to be, but not all groups do. Ray suggested: meeting dates and topics, reports of meetings, a continually updated list of newsletters and sigs, product reviews and a product guide.

A short discussion about email led to no conclusion as the people present had such widely varying skills and experience at handling mail. It was agreed that all really ought to know how to use it.

Curtis agreed to email a list of journals and their addresses and anything else he thought helpful to everyone with an email address.

Walter Spunde 'Point-wise Calculus'

reported by Anthony Camacho

Walter Spunde began by displaying a table that he showed at a workshop for high school maths teachers:

x	0	0.5	1
f	3	1	2
f'	1	-1.5	-2
g	2.5	3	1.5
g'	3	2	-2
d			
--(fg)			
dx			

He had asked the teachers what was the derivative and no one could do it. They forgot that the derivative has to be at a point and one of the teachers got close by beginning to draw a graph. One can plot a series of points and the slope at each and deduce a curve but there is a danger that the sample is inadequate. Given second or higher derivatives we can do better. The message was that by calculating derivatives one may get a more intuitive appreciation of functions and the rules of calculus.



Walter Spunde and Ken Iverson

The details of the argument are in the paper. By using APL it is possible to appreciate the functions in the paper; in any other language the point would be lost in lengthy listings. It was interesting to discover that it took 15-20 seconds to calculate 10th order derivatives at 100 points, which implies the calculation of all lesser derivatives, and so (if the sample is adequate) graphs of each derivative can be drawn, whereas to calculate the formula for the tenth derivative using Mathematica

took an hour and a half. This is not to complain Mathematica is slow: it can calculate the derivatives themselves fast enough; the point is that calculation of numerical results is sometimes a better way to appreciate a formula than symbolic manipulation. Nor is this to belittle symbolic manipulation. What it does is to distinguish clearly between the theory and the application of arithmetic.

Norman Thomson asked whether the function samples allowed for complex numbers and Walter invited him to help extend the domain of the arguments. Norman said he had done the equivalent.

Murray Eisenberg suggested some of the formulae were too difficult to show to the students. Walter agreed and remarked he had shown us more than he would show his class. Murray suggested that the limitations of Taylor should be covered and Walter agreed. He shows his students that Taylor's theorem needs to be proved.

Asked what graphing functions he uses Walter said he wrote his own, at first in I-APL and now in APL*PLUS. Alan Mayer asked whether he advocated teaching calculus without using the classical symbols. Walter replied that he had a government grant to do this as an educational experiment this year. The course covers the same ground as in High School, but it also includes enough symbolism to enable the students to use a symbolic manipulator such as Mathematica.

Pavel Luksha

'Modern Algebra Self-taught Through APL'

reported by Anthony Camacho

Pavel Luksha began by telling us that he was not a high school student but that he would be one if he lived in North America. The paper is about his learning of modern algebra. In his opinion self-teaching is preferable to straight teaching.



Bob Bernecky and Pavel Luksha

The mathematical disciplines require a powerful calculator and APL provides this. Pavel learned APL and modern algebra at the same time and found each helped the other. He used *Modern Algebra with Applications* by William J Gilbert as his text book and I-APL/Atari with Garry Helzer's *Encyclopedia of APL* for his APL, although it was slow. He found the book hard to begin with — he needed someone to guide him into it — but

after a while he began to understand how to translate its formulae into APL and then made rapid progress.

Pavel said that self-teaching required great will-power and self discipline, but that in any successful teaching there has to come a time when the teacher relinquishes control and lets the student fend for himself. With self teaching this comes right at the beginning and the effect is to increase the psychological reward of success — the feeling of achievement. Pavel then took us through his paper (see the Proceedings).

Pavel agreed with a questioner that Mathematica could help in analysis. He told Murray Eisenberg that he had learned APL at the same time as finite algebra. He was asked his age which is 16 [does anyone know of a younger presenter at any APL conference?] and he confirmed that he had studied mostly alone but later was able to interest a friend. Doing it together is more enjoyable. Finally Pavel told us that he was getting interested in catastrophe theory and hoped to write a paper about it.

Alexander Skomorokhov 'Adaptive Learning Networks in APL2'

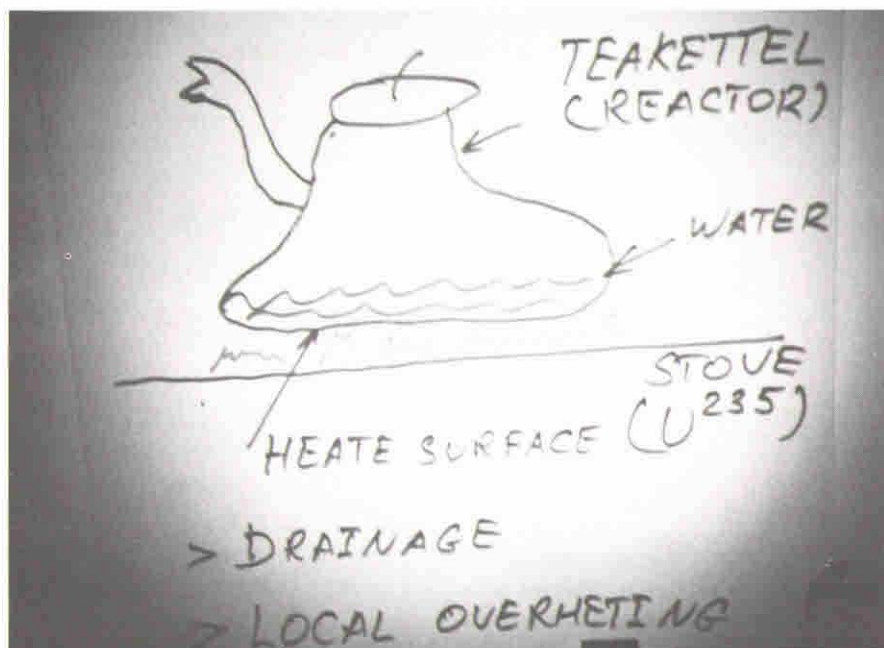
reported by Anthony Camacho

Sasha began by saying that computers are very limited and cannot yet be programmed to recognise a man or woman. His paper is about a less ambitious use, where an Adaptive Learning Network program can use the input and output measurements of a system to construct a network which models the system. The network may have many layers and elements in each layer with widely varying connections between each element in one layer and elements in the next layer down. The structure is not controlled by the programmer but is self-organised.



Dr Skomorokhov

He used as an example three ridiculous forecasts of the price of IBM stock and showed how, from the records of prices a forecast could be made with a layer of two-input elements each producing one input of the next layer of two-input elements. As he put it there are two questions: when to stop and how many Crays to use! He then gave the details of the process (all available in the paper). In essence the method tries all responses for all possible pairs of inputs and selects the best matches, adding layers until improvements in prediction accuracy are no longer obtained. From the resulting network the best predictors in the last layer are selected and then all elements and variables which do not contribute to them can be eliminated from the network and the useful part of the network reduced to a tree.



To convince us of the value of this, Sasha quoted the example of heat flux burnout of a nuclear reactor: what happens when you leave your kettle on the stove too long — it boils dry and then melts.

He concluded by saying that he had found the user-defined operators of APL2 extremely valuable.

L. Fraser Jackson

'Computer Assisted Timetabling Using J'

reported by Anthony Camacho

At Victoria University of Wellington they had a problem: they have the tightest possible fit between courses and space to hold them in, the timetable was last fully revised in 1965 and the multiple patching that had been necessary since then had resulted in many anomalies and inefficiencies. Subject class sizes have changed since 1965 and subjects have been added and dropped. The main problem is with the large classes. There are only nine large rooms and in 360 hours 96% of bookings could not get into smaller rooms. A previous attempt had begun with good intentions and asked departments for the restraints they would like to impose on the timetable: the result was a set of restraints which could not be met and no simple way to tell which should be least reluctantly dropped.

Professor Fraser Jackson adopted the policy that he would simply try to fit all the classes into rooms and avoid the more serious clashes. The process was handled in three steps:

1. Determine the type and size of room which would fit each class
2. Within type of room assign classes in decreasing order of size
3. Permute assignments to minimise change and remove any known infeasibility.

The result was a workable timetable. The system is used by people who don't know anything about J. Departmental secretaries have Macintoshes and University Administration uses only PCs; it was a great advantage to have a system that could be used in exactly the same way on either type of machine.

There are some further things that should be done. Too many advanced classes are held on Tuesday and Friday; they should be more evenly spread through the week. Some departments dislike having all their staff teaching at the same time. It would be welcome to give departments more say in their own timetabling.

The new timetable is to be introduced in 1994 and Professor Fraser Jackson hopes to extend it to timetabling the smaller classes in 1995.

Keith Smillie 'Rolling Dice: Some Notes on J and Teaching Probability'

reported by Anthony Camacho

Keith Smillie follows Ken Iverson in wanting to use APL and J to teach a wide range of subjects and in the wish to use just the parts of the notation necessary for the task in hand. As he said "Most computing science courses have all the charm and interest of a course in the conjugation of verbs". We were referred to a PostScript file on `ftp.cs.ualberta.ca` in `pub/smillie/intj.ps`, to *Lady Luck: The Theory of Probability* by Warren Weaver, to Kendall and Stewart *The Advanced Theory of Statistics Vol 1*, to Weldon's data in the 11th Edition of the *Encyclopaedia Britannica* and to an article on the Central Limit Theorem in *New Scientist* for 14 December 1991.

The presentation followed the paper quite closely. Keith used triangular dice to reduce the space taken by the displays but the functions work equally well for cubical or tetrahedral dice.



An APL Competition for Russian Schoolchildren:
The winner is seated in front of Dmitri Lukyanov

Stephen Jaffe 'Modelling Petrol Chemistry in the Era of Clean Fuels' (preceded by announcements)

reported by Anthony Camacho

The student helpers were thanked by Richard Procter. Sasha Skomorokhov won the prize for the most elegant Logo. Larry Moore was awarded a blue ribbon. Jon McGrew awarded Jim Brown the SigAPL Outstanding Achievement Award. Alain Delmotte, the programme chair of APL 94, urged us all to come to Antwerp in Belgium from 11 to 15 September 1994 and bring a non-APLer with us. Alain reconfirmed that APL 94 will be supported by the ACM.

Stephen Jaffe of Mobil Oil was on the organising committee of APL 87 at Dallas and holds two US patents expressed in APL.

He spoke about the problem of producing cleaner fuels. The demand is for high octane and the crude supply is mostly heavier, so the crude has to be split into lighter elements and as far as possible bad compounds should be converted to good ones, and then the worst elements of the lighter mixture have to be removed. Of course the end result is a range of oils, some of which are for fuel, some for diesel engines, some for aircraft (mainly jet engine fuel) and some for automobile fuel. The latter has to be high enough octane rating to avoid risk of damage to engines.

Once the analysis broke the mixture into four or ten sections, each of which was a mixture of a great many elements; now it is necessary to analyse about 3500 elements in the overall mixture. The analysis has to be fast and so the way it is done is by using APL to generate FORTRAN. About 10,000 lines of FORTRAN are generated.

The analysis is all done on a PC 80486 based machine running Dyalog under Windows. This caused some delegates surprise; it re-inforced one of the messages of APL 93 that there is no need to wait for new developments before you can write windows systems in APL. People are doing it now. Some have systems working, such as Mobil and the competitive advantage it gives is probably significant.

APL 93 Postscript: “Don’t Stop Thinking About Tomorrow”

by Adrian Smith

After APL 91, I upset a few people by wondering to myself if it had been a conference or a reunion. I am sorry, but the past of APL should not be of interest to APLers; let us leave that to the language historians and get on with designing the future.

Toronto 93 could hardly have been a greater contrast. We started with McIntyre showing us how the ancient Egyptians got stuck with an inefficient and redundant script because “it was the Language of the Gods” and change was therefore unthinkable. After 3,000 years, it was replaced by Greek. Typically, he left the conclusions hanging in the air, but the parallel between conventional mathematical notation (ugly, inconsistent, un-linearised, impossible to extend, but sacred) and APL (linear, consistent, extensible) was plain to us all.

From the conference satchels (“APL2 co-operates”) to the closing plenary, the accidental theme of APL 93 turned out to be the fit of APL into modern windows interfaces and networked environments. There was genuinely exciting stuff on view, both in the formal papers, at the Vendor forums, and at the Exhibition. I left APL 93 feeling more optimistic about the future of APL as a notation than I have felt for years.

Conference Highlights

- **The Exhibition.** This was spacious, accessible and well-attended. The technology was impressive too — it was fun to run through a demo at the Technosis stand which just happened to include networked access to an SQL database on one of Dyadic’s machines several booths away. The Soliton balloons were a real hit with the children, particularly when Richard’s had to be rescued from the ceiling by an athletic Dane.
- **GUI APLs.** Someone commented to me “this feels like Dyadic’s coming-out party”. But ISI and IBM were in there too! Dyadic’s beer-and-sandwiches Vendor forum ran out of food, and nearly ran out of beer; IBM also showed a very competent GUI APL2 under OS/2, and ISI had to re-run Eric Iverson’s “Window Driver” tutorial due to popular demand.
- **McIntyre on Egyptology.** I don’t care if this had anything to do with APL, I just like hearing Donald lecture!

- Gérard "*I am driven by Symmetry*" Langlet. Gérard has to be experienced — as a live show he is brilliant, even at 8.30am. Some of the images he showed were genuinely stunning, and I left the talk convinced that virus DNA must somehow implement $\neq \setminus$ in hardware!
- Jack Rudd on APL in satellite surveillance. In part for the retrospective, but more importantly for his view of a parallel future, and APL's role in it.
- Richard Levine's "*Birds of a Feather*" on APL utilities. The Toronto Toolkit is already a good product, and after an hour and a half's hard brainstorming, it should be even better.
- The Conference Literature. Everything from the initial flyers to the Proceedings was well-designed and professionally produced. I suspect that to have material of this quality 'up front' is worth at least 50 delegates; it certainly influenced me in signing up early.

The organisation was effective without being officious, and the security (if present at all) was far less overt than at Stanford. The site was almost ideal, with the exhibition area and the lecture rooms within seconds of each other (although some signposts would have been quite useful). The banquet was well up to standard — particularly the gardens — and (of course) Toronto fully lived up to its billing as one of the top conference cities in the world. Where else can you ride a streetcar for \$1.30 for a journey of several miles to an olympic-standard pool which is free? They even have districts called Scarborough, Malton and York!

Conference Lowlights

- The panel on GUI standards for APL. It was ridiculous not to invite MicroAPL (who started the whole thing 5 years ago on the Mac), and throughout the session, I felt a building sense of frustration in the audience (articulated in the end by Eke van Batenburg) that these damned implementors needed some users to knock their heads together! Maybe some good will come of it, but it left me with an uncomfortable feeling by the end.
- Please can we have water at coffee-breaks, on speakers' tables etc. (and indeed at the banquet). Toronto is warm in summer, and if you try to take all your fluid on board as coffee, you end up pickled (or at least I do).
- The Audio-visual stuff. In these hi-tech times, it should be quite normal to rock up to a podium with a Compaq portable, and expect to see some means of projecting it! Generally, a projector was found, but often after a good deal of scurrying about; in at least one case a paper was more or less wrecked when the organisation discovered that it required 3 projectors, and only possessed 2. Several papers were also shunted from room to room at short notice, I assume to fit in with projection requirements.

- The picnic. OK, Stanford was a hard act to follow (and so by the sound of it was St Petersburg), but the Algonquin Island site really was a bit of a grot-hole. The buses were cunningly timed to just miss the ferry (one per hour), and the site was hot, dusty and cramped, with no decent shade. The food was fine, but (as a final offering to the von Neumann paradigm?) the processing was strictly sequential and the queues consequently lengthy.
- the framed speaker's plaques — gimme a teeshirt every time!
- the loss of Richard's Soliton balloon to a tree mere yards from our hotel.

I also felt that perhaps the program was perhaps a little bit 'bunched'; all the stuff on user-interfaces and DDE was on the first day, and all the Statistics came together on the third. I assume that this was to tempt 'day' delegates who had a particular interest, but for the rest of us it had the effect of reducing the accessibility of the papers from our own interest group. However I do think it was good to run only two streams, with plenty of parallel slots for rerun tutorials and 'BOF' sessions. Not a serious quibble really, I'm sure that however carefully streams are arranged, someone has to make hard choices.



Gene McDonnell learns K from Arthur Whitney

GENERAL ARTICLES

This section of VECTOR is oriented towards readers who may neither know APL, nor be interested in learning it. However we hope you are curious about how, under the right conditions, such impressive results can emerge so quickly from APL programmers



View of "Casa Loma" from the gardens, taken at the APL 93 banquet

Minesweeper — GOTO Considered Futile

by *Adrian Smith*

Prerequisite

For those of you (surely a small minority) who haven't yet played the "Minesweeper" game that comes with Windows 3.1 — go and play it! If your boss tries to sack you for playing games during working hours, you can quite reasonably claim that you are actually acclimatising yourself to the Object-oriented paradigm. You can also (incidentally) try to beat Richard's best times of 89 secs (intermediate) and 289 secs for 'the big one'. When you think you have the hang of it, get out the APL GUI of your choice, and see if you can knock off a reasonable facsimile in your lunch hour.

Then do it without GOTO!

Making a State-Event Matrix (SEM)

The State-Event Matrix is the fundamental tool of the real-time programmer. It consists very simply of a three-column table, giving the actions to take when an object (in a given state) receives an event. Here is a tiny Minesweeper (sic) game in play, and the SEM as it stands at this point in the game:

```

1 0 null
1 0 null
1 0 null
0 1 bang
1 0 null
1 0 null
1 0 null
1 0 null
1 0 null
1 0 null
0 1 bang
0 0 prod
0 0 prod
0 1 bang
0 0 prod
0 0 prod

```



In this case I have used the first column to flag whether or not a cell has been investigated; the middle column marks the location of any mines; the third column is the action (function) associated with that state.

Here is the usual heap of `WC` junk to set up the layout, make some buttons, leave space for a status line, and call the rest:

```

∇ R+MS grid;sz;bw;map;bs
[1]  A Build MS form, then activate form.
[2]  grid+4 4[16 24]2pgrid
[3]  bs+24 ∘ sz+48 0+bs×grid ∘ bw+0.5×2÷sz
[4]  'map'WC'FORM' 'Mineswooper'(24 12)sz('COORD' 'PIXEL')
      ('SIZEABLE' 0)('MAXBUTTON' 0)('EVENT' 1001 1)
      ('BCOL' 192 192 192)
[5]  'map.RESET'WC'BUTTON' '&Reset'(0 0)(28 bw)
      ('EVENT' 30 'Reset')
[6]  'map.EXIT'WC'BUTTON' 'E&xit'(0 bw)(28 bw)
      ('EVENT' 30 'Quit')('CANCEL' 1)
[7]  'map.note'WC'LABEL' ''(28 4)(θ(2×bw))
      ('FONT' 'MS Sans Serif' 10)('BCOL' 192 192 192)
[8]  A Get ready to play ...
[9]  init_sem grid           A Starting State-Event Matrix
[10] Δnbr+ct_nb grid∅Δsem[;2] A Count problem neighbours
[11] ms_tab grid           A Draw 'em
[12] A All set, so here we go ...
[13] R=∅DQ'map'
[14] R+'Ready'
∇

```

This is where it all starts. As you can see, the probability is crudely hard-coded; a better simulation of the real game would be to chose a random pattern for a given number of mines. However this is harder!

```

∇ init_sem grid;nm
[1]  A State-event matrix records ...
[2]  A [;1] Explored (1|0)
[3]  A [;2] Live Mine (1|0)
[4]  A [;3] Action on Select Event (30)
[5]  nm×*/grid
[6]  Δsem+(nm,3)∅0
[7]  Δsem[;1]+0           A Start unexplored
[8]  Δsem[;2]+5=7nmp6     A 1 in 6 are live
[9]  Δsem[;3]+('prod' 'bang')[1+Δsem[;2]]
∇

```

Line[9] starts the process of coding the logic of the game into the state-event table. Dangerous squares go *bang*, safe ones must be *prodded* to discover their neighbour count.

Having seeded the playing space with our mines, we can then count them:

```

∇ ct←ct_nb mn
[1]  A Count mines in adjacent squares across grid.
[2]  mn←0,mn,0 o mn←0;mn;0
[3]  A Now shift every which way, and add up the planes ...
[4]  ct←+/ (1φ10mn) ; (10mn) ; (1φ10mn) ; (1φmn) ; (1φmn) ;
      (1φ10mn) ; (10mn) ; (1,0mn) ρ (1φ10mn)
[5]  ct←,1 1+1 1+ct
∇

```

... using an ancient line of code from my earliest *Life* program! Now we just need to knock out an array of buttons:

```

∇ ms_tab rc;nb;pos;bb;bp
[1]  A Make ms table as array of buttons ....
[2]  A bs is button size in pixels
[3]  'map.GRP' [WC'GROUP' ' '(44 0)(bs×grid)
[4]  nb←×/rc
[5]  bb←'map.GRP.B' bfm 1+1nb
[6]  bp←(nb,5) ρ c 10
[7]  pos←((nb,2) ρ bs) × qrc 1+1nb
[8]  bp[;1] ← c 'BUTTON'
[9]  bp[;2] ← ' ' A Button label
[10] bp[;3] ← +pos A Location in group
[11] bp[;4] ← c 2 ρ bs A Size in pixels
[12] bp[;5] ← c ('EVENT' 30 'plop')
[13] bb [WC'' + bp
∇

```

... and just for completeness ...

```

∇ fm←pfx bfm inx
[1]  A Make object name from prefix and inx
[2]  fm←'0123456789'[1+q10 10 10+inx]
[3]  fm←(c pfx),'' + fm
∇

```

That really is all we need to get an authentic Minesweeper facsimile on screen. Now the actions (*plop* is the 'switch' function which all the buttons operate when clicked):

```

∇ r←plop msg;inx;obj;action
[1]  A User clicked a square ... process event.
[2]  inx←1+1 3+obj+1=msg
[3]  A Get action from State-Event table ...
[4]  action←3→Δsem[inx;]
[5]  1'obj ',action,' inx'
[6]  r←1
∇

```

All the Logic is Here:

You will have to try this out to convince yourself! The fancy recursive behaviour that occurs when you clear a square with no neighbours ... the State-Event table holds it all!

When a 'safe' square is prodded, the first thing it must do is to record the fact, and then it changes its own action to null, so that if prodded again it responds by doing nothing! It checks for a 'won' game with a nifty 'not-equals' reduction, and 'turns over the tile' to reveal the neighbour count.

Now we encounter the single solitary \rightarrow in the entire workspace: it ducks out if it had any neighbouring mines! If not, it simply sends a 'Select' (30) event to all its neighbours, so that they in turn call *prod* (or *null* if they have already been cleared) and so on ad infinitum:

```

V obj prod inx;mn;nbrs;bb
[1]  A Prod safe squares ... see <bang> for the other sort
[2]  Δsem[inx;1]+1 ◊ obj □WS'STATE' 1
[3]  'map.note'□WS'CAPTION'((^/#!/Δsem[;1 2]));//Well Done Lad!')
[4]  A Change action to do nothing ...
[5]  Δsem[inx;3]+c'null'
[6]  obj □WS'CAPTION'(' 12345678'[1+Δnbr[inx]])
[7]  A Clear neighbours if zero count ...
[8]  →Δnbr[inx]+0      A <<<<<<<<<< SPOT the GOTO <<<<<<<<
[9]  nbrs+grid nbr inx
[10]  nbrs+(nbrse1ρΔnbr)/nbrs ◊ nbrs+nbrs-inx
[11]  bb+'map.GRP.B'bfm nbrs-1
[12]  □NQ"(c"bb)","30
V

V r+grid nbr inx
[1]  A Find all neighbours of inx on grid
[2]  r+8 2ρgridvinx-1
[3]  r+(8 2ρ-1 -1 -1 0 -1 1,0 -1 0 1,1 -1 1 0 1 1)+r
[4]  A Lose any that fall off the edges ...
[7]  r+1+gridiqr
V

V obj null inx
[1]  A Do nowt (quietly!) ...
[2]  obj □WS'STATE' 1
V

```

Something very similar happens when you tread on a mine:

```

▽ obj bang inx;bb
[1]  R Prod dangerous squares ... see <prod> for the other sort
[2]  Δsem[inx;1]+1 ◊ Δsem[;3]+c'show'
[3]  obj [WS'CAPTION' 'M'
[4]  'map.note'[WS'CAPTION' 'Oh bother ...'
[5]  R Clear rest of board ...
[6]  bb+'map.GRP.B'bfm ~1+;1+pΔsem
[7]  [NQ"(c"bb),~30
▽

```

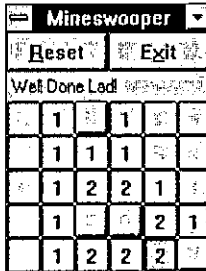
Every tile gets its action switched so that it simply turns itself over ... and then a Select is fired at the entire board to run 'em all:

```

▽ obj show inx;bb
[1]  R End of game ... expose square
[2]  Δsem[inx;1]+1 ◊ Δsem[inx;3]+c'null'
[3]  obj [WS'CAPTION'('-M'[1+Δsem[inx;2]])
▽

```

Just to prove it can be done ...



J Solution to Enigma 685

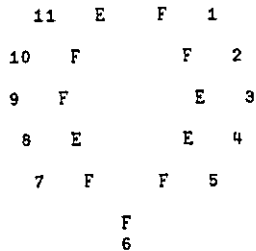
from New Scientist 1840, dated 26
September 1992

by Dave Ziemann

The Problem

The problem, set by Keith Austin, is as follows:

I came upon a clearing in the jungle. At the centre was a circular array of plates, some with food (F) and some empty (E). The array is shown in the diagram with the plates numbered for convenience:



My guide explained that the plates were for a food-and-fast ceremony over 11 days. The person fasting would select a food plate and eat its contents on the first day; for the second day, (s)he would move clockwise to the next plate and eat its contents, and so on; the first plate had to be a food plate but on some later days there would be empty plates which would produce fast days. The person fasting had to choose a starting plate so that at the end of each of the 11 days more food plates had been used than empty plates. For example, plate 1 is not a starting place as, at the end of day 4, 2 food and 2 empty plates would have been used. However plate 5 is a starting plate.

Question 1: Which plates are starting plates?

I then moved to a second clearing which again contained a circular array of plates for a 47-day ceremony. As I went round the plates were:

```

 1  2  3  4  5  6  7  8  9 10 11 12
 F  F  E  E  E  E  F  F  F  F  E  F

13 14 15 16 17 18 19 20 21 22 23 24
 E  F  E  F  E  F  F  F  F  E  E  E

25 26 27 28 29 30 31 32 33 34 35 36
 F  F  E  E  F  F  E  E  F  E  F  F

37 38 39 40 41 42 43 44 45 46 47
 E  F  E  F  F  F  F  E  F  F  E
    
```

Question 2: Which plates are starting plates in the second clearing?

I then moved to a third clearing which had a circular array of 45 plates which were the same as plates 1 to 45 in the second clearing. These plates were for a 45-day ceremony.

Question 3: Which plates are starting plates in the third clearing?

I then moved to a fourth clearing which had a circular array of 150 food plates and 140 empty plates; I forgot the order they were in. These plates were for a 290-day ceremony.

Question 4: How many starting plates were there in the fourth clearing?

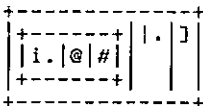
The Solution

We can start by defining the array of 11 plates as a boolean list in J:

```
p11=. 1 1 0 0 1 1 1 0 1 1 0
```

We want to be able to consider each plate in this list as a potential starting plate. So we can generate a table where each row contains p11 successively left-shifted by one place further, so that each place in turn acts as a starting plate. The fork as produces all the possible shifts of its argument list:

```
as=. i.@#|.]      NB. all shifts (monad)
as
```



```

      as p11
1 1 0 0 1 1 1 0 1 1 0
1 0 0 1 1 1 0 1 1 0 1
0 0 1 1 1 0 1 1 0 1 1
0 1 1 1 0 1 1 0 1 1 0
1 1 1 0 1 1 0 1 1 0 0
1 1 0 1 1 0 1 1 0 0 1
1 0 1 1 0 1 1 0 0 1 1
0 1 1 0 1 1 0 0 1 1 1
1 1 0 1 1 0 0 1 1 1 0
1 0 1 1 0 0 1 1 1 0 1
0 1 1 0 0 1 1 1 0 1 1

```

By default, J performs operations along the leading axis of an array. In what follows it will therefore be convenient to hold plate vectors as columns rather than rows. In this case a transpose is not necessary because the matrix is symmetrical.

A plate is only a starting plate if, for each and every day, the cumulative number of Food plates is greater than the cumulative number of Empty plates. We can build a monad called `cumas` which performs this cumulation as follows:

```

cum=. +/\      NB. cumulate (monad)
cumas=. cum@as NB. cumulate all shifts

```

```

      cumas p11
1 1 0 0 1 1 1 0 1 1 0
2 1 0 1 2 2 1 1 2 1 1
2 1 1 2 3 2 2 2 2 2 2
2 2 2 3 3 3 3 2 3 3 2
3 3 3 3 4 4 3 3 4 3 2
4 4 3 4 5 4 4 4 4 3 3
5 4 4 5 5 5 5 4 4 4 4
5 5 5 5 6 6 5 4 5 5 5
6 6 5 6 7 6 5 5 6 6 5
7 6 6 7 7 6 6 6 7 6 6
7 7 7 7 7 7 7 7 7 7 7

```

Now we have to apply this verb first to the list `p11` (which indicates the Food plates), and then to the negation of `p11` (which indicates the Empty plates), and then ensure that the former exceeds the latter.

This suggests the use of a fork, as follows:

```
mfptep=. cumas>cumas@-. NB. more food than empty plates
mfptep
+-----+
|cumas|>|+-----+| | | | |
|      | | |cumas|@|-.| |
|      | | |      | |
+-----+
+-----+
```

```
mfptep p11
1 1 0 0 1 1 1 0 1 1 0
1 0 0 0 1 1 0 0 1 0 0
1 0 0 1 1 1 1 1 1 1 1
0 0 0 1 1 1 1 0 1 1 0
1 1 1 1 1 1 1 1 1 1 0
1 1 0 1 1 1 1 1 1 0 0
1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 0 1 1 1
1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1
```

The "and-insert" along the leading axis then reveals if each plate is a starting plate or not:

```
isasp=. *./@mfptep NB. is a starting plate?
isasp
+-----+
|+-----+|@mfptep| | |
|*./|/|| |
|+-----+| |
+-----+
```

```
isasp p11
0 0 0 0 1 1 0 0 1 0 0
```

The numbers of the starting plates can be found by incrementing the zero-origin result of the utility verb where, which produces indices from a boolean list:

```
where=. #1.@# NB. indices of boolean list (monad)
sp=. >:@where@isasp NB. starting plates
sp p11
5 6 9
```

And hence Question 1 is answered.

Questions 2 and 3 can now also be answered by application of this verb to similar boolean lists:

```

      p47=. 1 1 0 0 0 0 1 1 1 1 0 1 0 1 0 1 0 1 1 1 1 0 0 0 1 1
0 0 1 1 0 0 1 0 1 1 0 1 0 1 1 1 1 0 1 1 0
      p45=. 45{.p47
      sp p47
7 8 9 18 35 40 41
      sp p45
7 8 9 18 35 40 41

```

Changing the order of the Food and Empty plates in p11 will affect which plates are starting plates, although we may surmise that it will not affect the number of starting plates. We can test this by counting the number of starting plates in a series of random permutations of the list p11, by using the very useful verb A. (atomic permute) as follows:

```

      (i.6) A. 'abc'      NB. atomic permute example
abc
acb
bac
bca
cab
cba

      (?!11) A. p11
1 1 1 0 1 0 1 1 1 0 0
      (?!11) A. p11
1 1 1 0 0 1 1 0 1 0 1
      (?!11) A. p11
1 1 0 1 1 0 1 0 1 1 0
      # sp (?!11) A. p11
3
      # sp (?!11) A. p11
3
      # sp (?!11) A. p11
3

```

It would seem that that the number of starting plates depends only upon the total number of Food and Empty plates in any given food-and-fast ceremony. We can therefore construct what we might call a "typical ceremony" to represent a ceremony of a given size.

The dyad `tc` takes the number of Food plates and the number of Empty plates as its respective arguments:

```

      tc=. ,#1 0"_      NB. typical ceremony
      +/p11
7
      7 tc 4
1 1 1 1 1 1 1 0 0 0 0

```

The verb is constructed from a fork whose three tines are `,` (append items), `#` (copy) and `1 0"_` (a verb which always derives the constant `1 0`). We can count the number of starting plates in a typical ceremony with the verb `nsp`, which takes similar arguments:

```

      nsp=. #@sp@tc      NB. number of starting plates
      7 nsp 4
3
      +/p47
27
      27 nsp 20
7
      150 nsp 140
10

```

Which is the answer to question 4. The number of starting plates in any ceremony would seem to be equal to the total number of Food plates minus the total number of Empty plates.

Proof

The following proof of this was given to me by Paul Chapman.

Consider an arbitrary ceremony containing any number of Food and Empty plates. There are three possibilities; all the plates are Empty plates, all the plates are Food plates, or there are both Empty plates and Food plates. In the first case there are no starting plates. In the second case all the plates are starting plates.

In the last case, there must be a pair of plates in the sequence Food, Empty. The Empty plate cannot be a starting plate (no Empty plate ever can). The Food plate also cannot be a starting plate, because by the end of the second day the total number of Food plates will not exceed the total number of Empty plates. As neither plate in this pair can be a starting plate, the sequence does not contribute to the total number of starting plates, and so it can be removed from the ceremony without affecting the result. Eliminating all such pairs will eventually result in a complete set of Empty plates or a complete set of Food plates, from which the number of starting plates is immediately determined.

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

Hackers' Corner: A Windows Task-Killer for APL	Duncan Pearson	126
At Play with J	Eugene E McDonnell	128
Migrating Mainframe APL2/TSO Applications to APL*PlusII/386	Allan Gay	130
A GDDM Simulation for APL*PlusII/386	Allan Gay	133

Accepted for Publication in Vector Vol.10 No.3:

The Ultimate Turing Test	Gérard Langlet
A Proposal for a Control Operator in APL	Denis Sampson
A VSAM Simulation for APL*PlusII/386	Allan Gay

Hackers' Corner: A Windows Task-Killer for APL

by *Duncan Pearson*

Adrian Smith came to me the other day and said "If I start a Windows program from Dyalog with `⊞CMD` how can I kill it again from within APL?". So we dived for the manuals and found out a few interesting things. First of all the `⊞CMD` function, when used to start a Windows task, returns a number.

```
+⊞CMD 'notepad.exe' 'minimised'
12345
```

"Aha!" say we, "this looks like a task handle. Why not send a Windows 'QUIT' message to the task and see if it kills itself." If you are a Windows task then getting a `WM_QUIT` is like being locked in a room with a bottle of whiskey and a loaded revolver and being told to get on with it. It looked like a good bet.

So we did. We used `⊞WA` to associate the `PostAppMessage` windows function

```
⊞WA 'I user.P16|PostAppMessage U U I I4'
```

and we posted the task a `WM_QUIT` (message number 18)

```
PostAppMessage 12345 18 0 0
```

Hello DOS prompt, goodbye Windows — "bummer!" said Pooh!

Apparently `⊞CMD` returns something called an `hinstance` which is not a task handle but an *instance handle*. So is there a way from one to the other? As a matter of fact there is, but you have to look for it. There is a pair of functions in `toolhelp.dll` with which one can go round all the tasks in the Windows event-queue collecting information from them. One of the bits of information is the `hinstance` and another is the `task-handle`.

We associate them thus:

```
⊞NA 'I toolhelp.P16|TaskFirst = {I4 U U U U I I I I I U C[10] I U}'
⊞NA 'I toolhelp.P16|TaskNext = {I4 U U U U I I I I I U C[10] I U}'
```

They are pretty much the same. Each one takes a horrible structure called a `TASKENTRY` and returns it (the = means take and return the argument). We pass `TaskFirst` an empty structure and it fills it with details of the first task in the

queue. We pass the filled structure to *TaskNext* and it replaces the information about the first task with information about the next task. We carry on calling *TaskNext* until we find a task with the *hinstance* that was returned, get the task handle of that task from the structure and send it a *WM_QUIT*.

What I omitted to say was what the horrible structure meant. We get back twelve numbers, a ten-character text vector and two more numbers. The text string is the task name, the second number is the task handle and the fourth is the *hinstance*. I'm sure the rest is very useful, but not right now so I will ignore it.

```

      v taskfns
[1]  n Associate some useful Windows fns
[2]  'pam'[]WA'I user.exe.P16|PostAppMessage U U I I4'
[3]  'first'[]WA'I toolhelp.P16|TaskFirst =(I4 U U U U I I I I I I U C[10] I U)'
[4]  'next'[]WA'I toolhelp.P16|TaskNext =(I4 U U U U I I I I I I U C[10] I U)'
      v

      v list+Task_List;qq;pam;first;next
[1]  n List handles and names of all extant tasks
[2]  taskfns o list+'
[3]  qq+2>first<(12+40),( <10p' '),0 0
[4]  Next:nm+13>qq o nm+(~1+nm:[]AV[1])+nm
[5]  list,+<nm
[6]  ->('DIALOG'=6+nm)+0
[7]  qq+2>next<qq
[8]  ->Next
      v

      v Kill_Name name;qq;pam;first;next
[1]  n Kill task by name
[2]  taskfns
[3]  qq+2>first<(12+40),( <10p' '),0 0
[4]  Next:->(name=(pname)+13>qq)+Kill
[5]  ->('DIALOG'=6+>13 qq)+0
[6]  qq+2>next<qq
[7]  ->Next
[8]  Kill:qq+pam(2>qq)18 0 0
      v

      v Kill_Num hinstance;qq;pam;first;next
[1]  n Kill by instance handle (returned by dyalog on []CMD)
[2]  n N.B. It had better be there!!
[3]  taskfns
[4]  qq+2>first<(12+40),( <10p' '),0 0
[5]  Next:->(hinstance=4>qq)+Kill
[6]  qq+2>next<qq
[7]  ->Next
[8]  Kill:qq+pam(2>qq)18 0 0
      v

      Kill_Name 'CLOCK'      n Harmless enough
      Kill_Name 'PROGMAN'   n Less sensible ...
      Kill_Name "Task_List" n ... for a very clear desktop!

```

At Play with J

by Eugene E McDonnell

I had a request recently from someone who wanted to apply a verb a different number of times to a list of arguments. What was wanted was a simpler way of writing:

```
(f a),(f f b),(f f f c)
```

My initial response was to say that J did not as yet have a way of describing Multiple Instruction-Multiple Data machine architectures (MIMD), although such a mechanism had been described [Be91]. I pointed out that a collapsing transpose could solve the problem, but my questioner would have none of that, as it implied a great deal of useless computation. There the matter rested for a while. After several months I had another request from the same person who wanted to know if I had made any progress on the problem. Actually, I hadn't thought about it at all in the interim, but since my questioner seemed to be a determined type, I gave it a few minutes more thought, and found what I think is a neat use of one of J's more interesting differences from APL, the way `scan` is defined: that is, the verb applied is monadic, not dyadic.

For example, whereas in APL one writes `+ \ 1 2 3` to obtain the continued sum of the values in the argument, in J one would have to write `+/ \ 1 2 3` to obtain the same result.

```
+ \ 1 2 3
1 0 0
1 2 0
1 2 3
```

Here the monadic verb `conjugate`, denoted by `+`, is being applied, first to `1`, next to `1 2`, and last to `1 2 3`; since these are real numbers, their conjugates are the same as the arguments, and since J reshapes results so that they conform, and then appends them, we get the zero fills at the right of the top two rows. Compare this with

```
+ / \ 1 2 3
1 3 6
```

which is the analog to APL's `+ \ 1 2 3`.

Finally, here is the solution to the MIMD problem.

First I define three variables, a, b, and c:

```
'abc'=.3 4 5
```

Next, I define a verb f to be the natural logarithm (\wedge).

```
f=. $\wedge$ .
```

and apply it once, twice, and thrice, to a, b, and c, respectively:

```
f a
1.09861
f f b
0.326634
f f f c
_0.742579
```

This is the desired result, but done the hard way. Now for the easy way:

Define a verb g which in which the verb f is applied (@) to the tail (t:) of its argument a number of times (\wedge ;) equal to the length (#) of its argument:

```
g=.f@{: $\wedge$ :#
```

For example, g 3 1 4 1 5 9 applies f six times to 9:

```
f f f f f f 9
0.854804j1.01575
g 3 1 4 1 5 9
0.854804j1.01575
```

Perhaps you already see how this will end. We apply the prefix scan (\) adverb to g, and apply this derived verb to a, b, c:

```
(g\ )a,b,c NB. apply g to successively longer prefixes
1.09861 0.326634 _0.742579
NB. q.e.f.
```

Showing once again that where there's a will there's a way. Note that because of the way prefix scan is defined, it is easy to visualize how, in a multiprocessor environment, the applications of g to all three arguments can be carried out simultaneously.

[Be91] Bernecky, Robert, and Roger Hui, 'Gerunds and Representations', APL Quote Quad 21, 4, August 1991, Stanford, California 1991, pp 39-45

Migrating Mainframe APL2/TSO Applications to APL*PlusII/386

by Allan Gay, Cocking & Drury Ltd.

Introduction

For the past few months, my colleague Kevin Ryall and I have been working on migrating a client's mainframe APL2/MVS applications into APL*PLUS II/386 on a 486 machine. Kevin has shouldered the lion's share of the drudgery while I have had all the fun. In this article I introduce some of the interesting aspects of the project. Subsequent articles will expand these.

I am indebted to Dave Ziemann for his very helpful critiques of this article and the others in this series. Any errors which remain are mine, not his.

Core Issue

The mainframe applications make extensive use of GDDM, VSAM, and APL2-isms, none of which are included in APL*PLUS II/386. This posed a major question right at the outset — substitute or simulate?

Substitution entails rewriting large sections of the apps to eliminate all references to unavailable facilities. For instance, the functions to build and operate a complex fullscreen panel via GDDM have to be completely rewritten to use APL*PLUS II/386's `□WGET`, `□WIN`, `□WKEY` and `□WPUT` facilities. Worse still, conceptual differences tend to enforce a different functional decomposition of the task. Before you know it, you can be embroiled in rewriting great swathes of code from scratch.

Simulation, in contrast, means adding a layer of new code whilst largely preserving the existing code and the interfaces. If the simulation is sufficiently comprehensive, migration can be relatively straightforward, although there will be a weight penalty.

Because the use of unavailable mainframe facilities was so extensive, the simulation option was chosen.

Simulation Form

Each simulation takes the form of a workspace which must be copied into the application workspace to supply a missing facility. Because defined functions are being substituted for intrinsic language elements such as shared variables, some recoding at the actual point of interface is unavoidable, but application logic is still preserved.

Which Simulations?

This is the first in a series of articles in which I'll be describing simulations of GDDM, VSAM, APL2 defined operators, and some other APL2-isms.

Looking at this list, you might be forgiven for thinking that APL*PLUS II/386 seemed pretty limited compared to mainframe APL2, but you'd be wrong. If we were migrating in the opposite direction, we'd find ourselves bemoaning the lack of *WIN*, the component filing system, and the Split and Mix functions.

All the simulations require APL*PLUS II/386 Release 4 or higher, operating in Evolution Level 2 mode.

Examples of Use

First, GDDM's ASDFMT operation. Here's how we do it on the mainframe:

```
CTLg+402,(0 1+pFLDS),,(11+pFLDS),FLDS
RES+FSchk CTLg
```

And here's how we do it on the PC —

```
RES+FSchk 1>GDEXEC 402,(0 1+pFLDS),,(11+pFLDS),FLDS
```

In essence, we replace two references to the shared variable by "1 GDEXEC". The important thing to note is that the actual data and the application logic by which it is developed remain exactly the same.

Next, let's look at how we replace a record on a VSAM key-sequenced dataset (KSDS). On the mainframe, we do —

```
DATV+RECORD           A Put data into shared-variable.
CTLV+'W'              A Issue the WRITE request.
VSAMchk CTLV         A Check the return code.
```

And on the PC we do:

```
VSAMchk 1>VSKSW RECORD      * Write the record.
```

In this case, the defined function *VSKSW* replaces the shared variable references.

Now let's turn to APL2 defined operators. In mainframe APL2, we say:

```
ALPHA +defined_op OMEGA
```

but in APL*PLUS II/386 we say

```
(ALPHA '÷') defined_op OMEGA
```

Things have got a little more hairy here because *defined_op* has now become a defined function. Therefore we have to bundle its input function and one of the inputs to the resulting derived function as a nested array.

Finally, some axial arithmetic. Mainframe APL2's

```
A+[1]3 3p16
```

becomes

```
A plus_with_axis 1,c3 3p16
```

and, again we have had to bundle one of the inputs with the axis specification.

Conclusion

That was just a taster but you can see that use of the simulations does not entail any violent upheavals in the original code. Using them, we're happily operating fullscreen apps with thirty-plus panels — many built on the fly — to front 7000-record KSDSs which have multiple indexes. The simulated defined operators are stacked n-deep, and our arithmetic is truly axial.

A GDDM Simulation for APL*PlusII/386

by Allan Gay, Cocking & Drury Ltd

Introduction

This article describes the GDDM simulation developed during the migration of some APL2/TSO mainframe applications to APL*PLUS II/386 on 486 machines.

As currently configured, the simulation can run complex character-based dialogue screens for data capture and display, and make them look very nearly the same as they would on a mainframe colour terminal.

Converting migrated mainframe code to use the simulation is straightforward.

Of the mainframe repertoire, the simulation provides 42 GDDM operations (see [GDDMREF]) and 2 AP126 operations (see [VSAPLTUG]). This may not seem much but it gives us an extensive range of character-based operations, including support for multiple GDDM pages and page-switching. Among the omissions are support for multiple pairs of shared variables and all the graphics.

The simulation's simple architecture makes it easy to bolt on more simulated GDDM operations, although the difficulty of actually simulating a given operation necessarily depends upon the nature of the operation concerned.

Interfacing the Code

The simulation has been geared to minimise the disruption entailed in converting mainframe code. An executor function named *GDEXEC* replaces AP126. Its argument and result variables assume the roles of AP126's shared variables. Input values are the same as for AP126 and, with some minor exceptions, output values are too.

GDEXEC's optional left argument receives any character data which would be assigned to AP126's *DAT* shared variable. *GDEXEC*'s right argument receives the integer data which would be assigned to AP126's *CTL* shared variable.

The result from *GDEXEC* is a 2-item vector. The first item contains the simulated AP126 *CTL* response, and the second item contains the simulated *DAT* response. Results are mostly identical to their mainframe equivalents. In the case of some

of the more recondite GDDM calls (e.g. *FSQSYS*), contents may diverge owing to platform differences, but result formats are still preserved.

Coding a Simulated GDDM Call

Here's a sample mainframe *ASCGET* to obtain the content of a specified field:

```

CTLs+422, fldno, fldlen      R Issue ASCGET
discard+CHECK CTLs         R Check retcodes
chars+DATs                 R Get field's content

```

And here's the simulated equivalent:

```

(nums chars)+GDEXEC 422, fldno, fldlen  R Do ASCGET & get rsit
discard+CHECK nums                    R Check retcodes.

```

Because the *ASCGET* operation requires no character input, AP126's *DAT* variable would not be assigned, so no left argument was supplied to *GDEXEC*.

Now here's a sample mainframe *ASCPUT* to write some text into a field. Because we are sending data to GDDM rather than receiving it, our subsequent processing is rather more perfunctory than in the previous example:

```

DATs+text                    R Post the field text
CTLs+424, fldno, ptext       R Issue ASCPUT for the field
discard+CHECK CTLs          R Check the retcodes.

```

And here's the same thing, using the simulation:

```

discard+CHECK 1> text GDEXEC 424, fldno, ptext

```

Batching Simulated GDDM calls

Both mainframe AP126 and the simulation allow you to batch multiple calls into a single exchange. This improves performance because the shared variable overhead is minimised. The calls' inputs are catenated together and assigned to the shared variables in one shot. The results are received en masse and separated afterwards. Each result has its own GDDM returncodes.

Batching is not difficult to simulate. In the inputs to the individual GDDM operations, data lengths always either are supplied as parameters or are implied by the nature of the operations themselves. Consequently, like AP126, the *GDEXEC* function is able to unbatch each operation's inputs, invoke the relevant simulated operation, and batch up the results.

Easy Conversion

In our migration project, we found that we could convert a migrated mainframe workspace with just a couple of hours' prosaic hoovering of the shared-variable uses. Typical applications operated between twenty and thirty display panels, most of which would be stashed in separate GDDM pages for speedy recall via *FSPSEL*.

The project ran for roughly nine months, during which time the simulation repertoire was extended and its behaviour enhanced several times, but the actual conversion of each application's GDDM code took no more time than this.

Architecture

The simulation comprises three layers of APL functions. The topmost layer consists of the three public functions: *GDEXEC*, *GDinit* and *GDterm*. The second layer consists of a validator function and an enactor function for each simulated GDDM operation. The third layer consists of utility functions. All the functions in the second and third layers are private.

Second-layer functions all bear a name commencing "GDv" (the validators) or "GDo" (the operations themselves). The trailing characters of the name are the numerical GDDM opcode. Thus, *GDo101* is the *ASREAD* operation (GDDM opcode 101) and *GDv424* is the operands validator for *ASCPUT* (opcode 424). This naming scheme enables *GDEXEC* to build the names of pre-existing functions dynamically, using opcodes from incoming parameter data. The functions are invoked via the Execute primitive.

AP126 operations, distinguished from GDDM operations by their negative opcodes, are dynamically renumbered into a 9xxx series. Thus incoming opcode -8 ("Query Modified Subset of Fields") is converted by *GDEXEC* to pseudo-opcode 9008, from which a function name may safely be formed.

Third layer functions bear 6-character names commencing "GD" and completed with a 4-character acronym intended to denote the purpose. For example, *GDhues* computes equivalent PC video attributes for the rows of a 17-column mainframe format matrix.

These very restrictive naming conventions facilitate code management and reduce the likelihood of name clashes against migrating applications. It would have been nice to have used the GDDM operation names (*ASREAD*, *FSQUERY*, etc.) to name the functions, but the authors of the code we were migrating had already pinched that idea some decades earlier.

Repertoire

In addition to API26's "Query GDDM Calls" and "Query Modified Subset of Fields" operations, the following GDDM calls are currently offered by the simulation:

ASCCOL	Specify Character Colours Within a Field
ASCGET	Get Field Contents
ASCHLT	Specify Character Highlights Within a Field
ASCPUT	Specify Field Contents
ASDFLD	Define a Single Field
ASDFLT	Set Default Field Attributes
ASDFMT	Define Alphanumeric Fields, Deleting All Existing Fields
ASFCLR	Clear Fields
ASFCOL	Define Field Colour
ASFCUR	Position the Cursor
ASFEND	Define Field End Attribute
ASFHLT	Define Field Highlighting
ASFIND	Define Input Null-To-Blank Conversion
ASFINT	Define Field Intensity
ASFMOD	Change Field Status
ASFYTP	Define Field Type
ASQCOL	Query Character Colours for a Field
ASQCUR	Query Cursor Position
ASQFLD	Query Field Attributes
ASQHLT	Query Character Highlights for a Field
ASQMAX	Query the Number of Fields
ASQMOD	Query Modified Fields
ASRATT	Define Field Attributes
ASREAD	Device Output/Input
ASRFMT	Define Multiple Fields without Deleting Existing Fields
FSALRM	Sound the Alarm
FSCOPY	Send Page to Alternate Device
FSFRCE	Update the Display
FSPCLR	Clear the Current Page
FSPCRT	Create a Page
FSPDEL	Delete a Page
FSPQRY	Query Specified Page
FSPSEL	Select a Page
FSQCPG	Query Current Page Identifier
FSQDEV	Query Device Characteristics
FSQERR	Query Last Error
FSQUPG	Query Unique Page Identifier
FSQURY	Query Device Characteristics
FSQSYS	Query Systems Environment
FSQWIN	Query Page Window
PTSQRY	Query Partition Set Attributes
PTSSEL	Select a Partition Set

The two partition set operations are dummies — created only because they were used by the applications we were migrating. They assume that there is a single extant partition-set and that its id is zero.

Data Structures

When the simulation is active, working global variables hold the simulated GDDM pages. Before *GDEXEC* can be used, the *GDinit* function must be run to start these global variables. At application shutdown when all GDDM simulation activity has concluded, the *GDterm* function may be used to revert the display to its pre-application state and to expunge the working global variables. *GDinit* and *GDterm* are thus in some degree analogous to *□SVO* and *□SVR*. They are nladic functions.

The working global variables are vectors, most of which are nested. Each element of a given vector contains data for one aspect of one extant GDDM page. Elements are added when new pages are *FSPCRT*'d. To save space, elements are nullified when their corresponding pages are *FSPDEL*'d. This is achieved by assigning an empty vector to such elements.

An extant page comprises: an alarm boolean, a rank-3 *□WPUT* screen-image, a cursor position vector, a GDDM format matrix, an equivalent *□WIN* matrix, a field ids vector, a boolean vector which flags modified fields, a default field-attributes vector, and a unique page id-number.

The simulated GDDM operations act upon these elements. The changes become visible when the *ASREAD* operation (opcode 101) is invoked to update the display and to capture keyboard input. For entertainment, a listing of the *GDo101* function is appended to this article.

Tuning

Simulation behaviour may be tuned by altering the values of certain manifest constants which are held permanently as global variables.

Tunable features include numeric checking (which may be set on or off to match your mainframe terminal), high-minus conversion (to hyphen if desired), and default video attribute (all unmapped screen areas are defaulted to low green on low black, but this can be changed).

The Keyboard

Mainframe layout has been retained as far as possible. It is supplemented by the APL*PLUS II/386 repertoire documented in the discussion of `□WKEY` in `□IREF`.

As on the mainframe, the cursor movement keys do not constrain the cursor to remain only in the defined fields. This apparent infringement of the APL*PLUS II/386 `□WIN` envelope is managed by treating these keys as exit keys, handing control to a utility function which operates the entire screen as one field. When the user steers the cursor back into a mapped field, the utility function cedes control and normal service resumes. On a 486 machine, we can get away with this sort of overhead, and my 25MHz 80386DX machine copes quite well, too. When teaching the camel to play the violin, some expenditure is necessarily incurred.

Tabkey behaviour is not identical to mainframe tabkey behaviour. When a tabkey is struck, the cursor moves to the start of the next geographical field. (A mainframe tabkey moves the cursor to the next geographical field row.)

Output-only fields, used for selection by cursor position when a PFkey is struck, are declared as "protected alphanumeric; immediate pen-selectable" by coding fieldtype 3 in the GDDM format matrix. When the corresponding `□WIN` format matrix is generated, bogus fieldtype 4 "PFkey selectable") is assigned. The `GD0101` function includes special logic to handle fields which have this fieldtype.

Video Attributes

Some compromise was entailed in reconciling the dissimilar video attributes of the mainframe terminal and the PC. The colours differ in hue of course, but APL*PLUS II/386's `SETPALETTE` function offers some scope for tuning PC colours. Except in the case of reverse-video fields, the simulation uses low-intensity black as the background colour; this is a tunable option.

The GDDM format matrix permits the specification of four attributes which influence the appearance of a field. They are Type, Intensity, Colour, and Highlight. Of these, Type influences only the default colour which is assigned when the Colour attribute is omitted.

The Intensity attribute may specify invisible, low, or high. Invisible is implemented as low-intensity black foreground symbols on a low-intensity black background. The other two intensities are implemented as foreground intensities for the colour concerned.

The two platforms have different Colour repertoires, but no insuperable difficulty presented itself. Mainframe pink became magenta, mainframe turquoise became cyan, and mainframe "default" became black.

The Highlight attribute gave the most trouble. The mainframe provides low, blink, reverse, and underscore. APL*PLUS II/386 does not support underscore on the PC colour monitor, so this attribute is simply ignored where coded. Blink is an alternative to background intensity, and was selected via 1 `POKE 902`. Reverse was implemented by exchanging the foreground and background colour numbers.

Size

Copying the simulation workspace into an application workspace reduces `WA` by 213KB. Across three applications operating a total of 80 GDDM pages, the additional space requirement per extant GDDM page averaged a little over 9KB.

Conclusion

There was a time, a few years ago, when you couldn't write a line of code for a full screen interface at Cocking & Drury Ltd without fifteen APLers interrupting to argue the toss over every line. Finishing anything was hell.

These days, all the action seems to be in GUIs. Freed of interruptions, I was able to do it up brown. But is there anybody still out there?

Next: the VSAM simulation.

References

[GDDMREF] . . Graphical Data Display Manager Base: Programming Reference, IBM, SC33-0101

[VSAPLTUG] . . VSAPL for TSO: Terminal User's Guide, IBM, SH20-9180

[IIREF] APL*PLUS II/386 Reference Manual, Manugistics Inc.

Appendix: the ASREAD function

The following example requires APL*PLUS II/386 Release 4.0 or higher, operating in Evolution Level 2 mode.

```

ΔZ+ΔA GDO101 ΔW;ΔC;ΔF;ΔG;ΔM;ΔN;Δp;ΔP;ΔQ;ΔR;ΔX;ΠIO
R GDDM simulation - opcode 101 - ASREAD service - PRIVATE
R α w are ignored.
R + is 2-item vector:
R 1> is 7-integer vector:
R [1-2] are retcode
R [3-4] are zero, denoting no results from this operation
R [5 6] are exitkey-class and -type
R [7] is count of modified fields.
R 2> is empty text vector.

ΠIO+1
ΔF+gdpage<gdpgflf
ΔF[(ΔF[;5]=2)/1+ρΔF;5]+1+KKgdnf
ΔF+ΔF[ΔG+ΔF[1 2];]
ΔN+gdpage<gdpgfn
ΔN+ΔN[ΔG]
ΔX+ΔF[;5]×0
ΔM+(1+ρΔF)ρ0
ΔC+GDcurv
ΔP+Δp+(cKKgdsz) Gdmapf" c[2]ΔX/ΔF

R PRELIMINARIES:
R Get format-matrix.
R Numfld checking enable/disable
R Sort it geographically.
R Get field-ids.
R Sort them geographically.
R Note keyable fields.
R Start off fld-modified flags.
R Get useable curpos for ΠWIN.
R Get maps of field positions.
R DIALOGUE PREAMBLE:
R Display updated screen.
R Turn off 'X SYSTEM'.
R If alarm request posted,
R honk the hooter and
R unpost the request.
R CONDUCT FULL DIALOGUE:
R If keyable fields are extant
R and cursor is in a field,
R conduct full dialogue,
R note any modified fields,
R & note the cursor position.

ΠWPUT gdpage<gdpgsfi
θ GDstat θp' '
+(-gdpgal[gdpage])/DDD
ΠARBOU 5p~1+ΠAV;ΠTCBEL
gdpgal[gdpage]+0
DDD:
+(-1cΔX)/FFF
+(0=ΔC[1])/FFF
ΔR+ΔC ΠWIN ΔX/ΔF
ΔM+ΔMvΔX\5+ΔR
ΔC+ΔR[3 4 5]
+HHH

R AWAIT EXITKEY WITHOUT INPUT:
R Just wait for an exit key.
R Note curpos within page.
R HANDLE ANY KEYBOARD-TOGGLE:
R If keyboard-toggle hit -
R . toggle keyboard state
R . update status line.

FFF:
ΔR+(1,1+ΔC) ΠWIN 0 0 24 80 32
ΔC+0,ΔR[4 5]
HHH:
+(ΔR[2]*KKgdAb)/JJJ
ΠSEG+θ 0 ΔQ+(-θΠ1[ΠPEEK 118])ΠPOKE 118
GDstat ''
+DDD

R HANDLE CURSOR ⇆⇆ KEYS:
R If cursor ⇆⇆ struck -
R . run special handler
R . update status line.

JJJ:
+(-ΔR[2]c393 420 391 389)/SSS
ΔC+(ΔR ΔP Δp) Gdmvcr ΔX/ΔF
GDstat ''
+DDD

```

```

SSS:
+(&R[2]*KKgdik)/VVV
[]SEG+θ θ ΔQ+4 2 4 2[0 1 2 4;[]PEEK 312]
[]SEG+θ θ ΔQ+ΔQ []POKE 312
GDstat ' '
→DDD
VVV:
[]SEG+θ θ ΔQ+2 []POKE 312
θ GDstat 'X SYSTEM'
gdpgsi[gdpage]+<[]WGET 3
→(θ=ΔC[1])/XXX
ΔC[1]+ΔC[1]<ΔX/ΔN
XXX: gdpgcu[gdpage]+<ΔC
gdpgcu[gdpage]+<GDcurs 1
gdpgmf[gdpage]+<ΔM[ΔG:1;θΔG]
ΔZ+0 0 3 0
ΔZ+ΔZ, KKgdkn[θ;KKgdex;ΔR[2];]
ΔZ+ΔZ,+/ΔM
ΔZ+(<ΔZ),<' '

```

R HANDLE ANY INSERT-TOGGLE:
 R If Insert-key hit -
 R make new poke value to
 R toggle insert/replace mode
 R & update the status line.

 R DIALOGUE EPILOGUE:
 R Turn off insert/replace mode.
 R Turn 'X SYSTEM' back on.
 R Store screen image.
 R Convert cursorfld index to
 R fldid (if not page-coords).
 R Store GDDM curpos, in
 R fld-coords if possible.
 R Deorder & store modflds bits.
 R Set retcodes and count.
 R Set GDDM exitkey-class & -no.
 R Set count of modified fields.
 R Wrap up result.

BAA-GUI: Advance Booking Form

Due to the interactive nature of the workshop it will be necessary to restrict the number of delegates in each stream. We are therefore giving an opportunity to reserve places.

Name: _____

Address: _____

Telephone Number: _____

Subject to demand we plan to run parallel streams in each of the main interpreters with a graphical user interface on the PC or Apple Macintosh. Please indicate below how many places you wish to reserve on each stream.

APL*PLUS II _____
 Dyalog APL/W _____
 APL.68000 _____
 APLIWIN/JWIN _____

Please return the completed form to:
 Duncan Pearson
 143 Hull Road
 YORK YO1 3JX

Preparing Articles for Vector

by Adrian Smith (Vector Production)

Background

Until further notice, Vector will be made up in MS Word (for DOS) and printed on my DeskJet with GoScript. Increasingly, material is arriving in Winword (or Windows Write) format, both of which Word reads with no problem (in fact Microsoft obviously used the Word format unchanged for Write — I just open the .WRI directly).

How you can help ...

To make my life as easy as possible, please observe the following guidelines:

- enter APL code with the Vector APL typewriter. This maps everything you enter to the VectorAPL truetype font, which has the same (STSC) mappings as I use in Word. It also makes getting listings etc. from Dyalog APL to Word a breeze (see my Mineswooper article on page 114).
- avoid all fancy characters. Don't use bullets (I suggest <tab>-hyphen-<tab> for indents like this), and don't use the typesetting quotes. I have a handy 'quote-inverter' to deal with these; if you put them in in WinWord I get × for opening quotes and Δ for closing. Not helpful!
- do not use <Insert,Symbol> to put in odd APL characters. I get a string of garbage across to Word (roughly what you see when you have <show field codes> on). This one really messes me up, and there is no easy way around it.
- pasted-in bitmaps are fine, but please please do not use embedded objects! These are an absolute pain to extract, and I usually end up printing the page from WinWord and taking scissors and glue to the finished Vector. If you don't want your diagrams stuck in at funny angles don't do it!

That really covers it. If you have Word for DOS, I would be delighted to send you a copy of the Vector style-sheet, so that you can submit articles 'ready to run' and save me even more work. If you are working under Windows, please send me a blank formatted HD disk, and I shall promptly return it with a Vector APL typebox and a runtime copy of Dyalog APL to drive it. The VectorAPL font (and indeed the typebox) are public domain, so give them away to your friends.

Index to Advertisers

Dyadic Systems Ltd	6
Manugistics	2
MicroAPL	44
International Travel Solutions Ltd	16
APL Booklist	32

All queries regarding advertising in VECTOR should be made to Gill Smith, at 04393-385.

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: 0272-730036

Authors wishing to use **Word for Windows** or **Windows Write** should contact Vector Production for a copy of the *VectorAPL* 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. Tel: 04393-385 (any time).

BAA: Membership Application Form

Membership of the British APL Association is open to anyone interested in APL. The membership year runs from 1st May to 30th April.

Name: _____
 Address Line 1: _____
 Address Line 2: _____
 Address Line 3: _____
 Post or zip code: _____
 Country: _____
 Telephone Number: _____

Membership category (please tick box): 93/94

(these rates also apply to renewals)

- UK private membership £12
- Overseas private membership £20
- Airmail supplement (not needed for Europe) £8
- Corporate membership £100
- Corporate membership overseas £155
- Sustaining membership £430
- Non-voting UK member (student/OAP/unemployed only) £6

PAYMENT – in Sterling 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 or Visa number.

I authorise you to debit my Visa/Mastercard 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

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.

1993/94 Committee

Chairman:	David Eastwood 071-922 8866 microapl@applelink.apple.com	MicroAPL Ltd., South Bank Technopark, 90 London Road, LONDON SE1 6LN
Secretary:	Duncan Pearson 0904-603510 100265.1564@compuserve.com	143 Hull Road, YORK YO1 3JX
Treasurer:	Nicholas Small 081-980 7870	8 Cardigan Road, LONDON E3 5HU
Journal Editor:	Anthony Camacho 0272-730036 acamacho@cix.compulink.co.uk	11 Auburn Road, Redland, BRISTOL, BS6 6LS
Activities:	Vacant Post	
Education:	Dr Alan Mayer 0792-295296 a.d.mayer@swansea.ac.uk	European Business Management School, Swansea University, Singleton Park, SWANSEA SA2 8PP
Technical:	Jonathan Barman 0488-648575 100116.1030@compuserve.com	Hill Top House, East Garston, NEWBURY, Berks RG16 7HD
Projects:	George MacLeod 0442-878065	Greymanle Associates Ltd., Bartrum House, Ravens Lane, BERKHAMSTED, Herts HP4 2DY
Publicity:	Misha Jovanovic 0753-853141	99 Oxford Road, WINDSOR, Berks SL4 5DX
Recruitment:	Mark Harris 0438-310155 kestrel@apl.demon.co.uk	Kestrel Consulting, Business & Technology Centre, Bessmer Drive, STEVENAGE, Herts SG1 2DX
Administration:	Rowena Small 081-980 7870	8 Cardigan Road, LONDON E3 5HU

Journal Working Group

Editor:	Anthony Camacho	0272-730036
Production:	Adrian & Gill Smith	Brook House, Gilling East, YORK (04393-385)
Advertising:	Gill Smith	Brook House, Gilling East, YORK (04393-385)
Support Team:	Jonathan & Bridget Barman (0488648-575), Ray Cannon (0252-874697), Richard Weber (0302-539761), Sylvia Camacho, Duncan Pearson, John Searle, David Ziemann (071-267 8032), 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.

Compass R&D Ltd
15 Frederick Sanger Road
Surrey Research Park
GULLDFORD, Surrey GU2 5YD
Tel:0483-302249
Fax:0483-302279

HMW Trading Systems Ltd
Hamilton House,
1 Temple Avenue,
LONDON EC4Y 0HA
Tel:071-353 4212
Fax:071-353 3325

Soliton Associates Ltd
Great Blankenberg 53
1082 AC Amsterdam
Netherlands
Tel:+31 20 570 8733
Fax:+31 20 570 8758
Email:jlh@soliton.com

Manuqistics
2115 East Jefferson St
Rockville
MARYLAND 20852 USA
Tel: (301) 984-5412
Fax: (301) 984-5094

Dyadic Systems Ltd
Riverside View, Basing Road,
Old Basing, BASINGSTOKE,
Hants. RG24 0AL
Tel:0256-811125
Fax:0256-811130

MicroAPL Ltd
South Bank Technopark
90 London Road
LONDON SE1 6LN
Tel:071-922 8866
Fax:071-928 1006

Kestrel Consulting
Business & Technology Centre
Bessemer Drive
STEVENAGE, Herts SG1 2DX
Tel:0438-310155
Fax:0438-310131
Email: kestrel@apl.demon.co.uk

Dutch APL Association
Postbus 1341
3430BH Nieuwegein
Netherlands
Tel:03474-2337