VisualAPL from APL2000.COM

What is APL?
APL is a computer programming language used to develop application systems. For more information see:
http://en.wikipedia.org/wiki/APL_(programming_language)

What is VisualAPL?
VisualAPL is an implementation of APL based on the Microsoft .Net framework, designed to make available the unique advantages of APL programming techniques, such as functional algorithm description and array-based analysis, to mainstream programmers.

VisualAPL is the next generation of APL, for the mainstream programmer as well as the domain expert.

Mainstream programming environments represent a significant business opportunity. Complex algorithms and business rules which are easily expressed in VisualAPL can be verified by domain experts who use VisualAPL. Fully-managed VisualAPL .Net assemblies can then be easily and directly incorporated into mainstream programming solutions. Technical resources and time are conserved and the GUI and data base development tools of the mainstream environment are used to create a superior software product using a multi-programming-language solution approach.

Is VisualAPL a .Net programming language?
Yes. VisualAPL is a .Net language just like C# or VB.Net, producing fully-managed code and compiled .Net assemblies which operate in a mal-ware-free virtual machine. For more information about 'managed code' go to:
http://en.wikipedia.org/wiki/Managed_code .

Can VisualAPL programs be used interchangeably in a .Net solution?
Yes. A .Net 'solution' is an application system developed using inter-operable 'projects' each of which may be programmed using a specific .Net language. VisualAPL can be used to create one or more component 'projects' in a .Net 'solutions.

Since .Net 'solutions' can incorporate multi-language components, the system designer can select programming languages for each component based on the language strengths and features. APL's strengths lie in algorithm expression and array-based data analysis, so it is often selected for business rules and calculation components of an application system solution.

Does VisualAPL implement object-oriented techniques?
Yes. VisualAPL is fully-object oriented. All VisualAPL programming entities are objects. Namespaces, classes and types defined in VisualAPL can be used by

any other .Net language. VisualAPL can use the namespaces, classes and types defined by any other .Net programming language.

Does VisualAPL support Unicode?
Yes. As a .Net language, VisualAPL is fully Unicode-based. All VisualAPL source code is expressed in Unicode text files.

Are VisualAPL programs compiled?
Yes, if desired by the system designer or programmer. Compiled .Net 'assemblies' (.exe or .dll) can be created with VisualAPL and are fully inter-operable with any other .Net language.

Compilation of VisualAPL source code is invoked in a manner identical to that of other .Net languages. .Net languages may be compiled using the Microsoft 'just-in-time' compiler from within Microsoft Visual Studio or may be compiled using the Microsoft 'command-line' compiler, provided with the .Net SDK, instead of using Visual Studio.

Compilation of VisualAPL source code to .Net assemblies is optional and up to the solution designer. VisualAPL source code can also be used in 'script' form which is analogous to traditional APL programming. VisualAPL scripts can be executed from within any .Net assembly, including those created using C#, VB.Net and VisualAPL. Whenever desired by the programmer, the VisualAPL source code in a VisualAPL script can be used to create a complied .Net assembly.

Are VisualAPL programs interpreted?
No. Traditional APL implementations rely on a proprietary, unmanaged-code, statement-by-statement interpreter. No such interpreter is required by VisualAPL.

This VisualAPL design principle means improved performance and extended deployment opportunities for VisualAPL solutions.

An interpreter is not necessary to provide a fully-interactive environment. The VisualAPL programmer has available the Cielo Explorer which provides a fully-interactive programming environment in which the result of each program statement can be visualized and analyzed by the programmer.

Does the VisualAPL programmer use a 'workspace'?
VisualAPL projects are not limited to the memory available in a traditional APL 'workspace'. As a .Net language VisualAPL uses the native memory management of Microsoft .Net which inherently supports both 32-bit and 64-bit memory address space of the hardware used to deploy the solution.

If desired by the VisualAPL programmer, a Cielo Explorer 'script' can be used to contain data and VisualAPL or C# source code. This programming mode is analogous to the 'workspace' mode of traditional APL implementations.

Unlike traditional APL implementations which mix data and code in a proprietary workspace format, VisualAPL source code is maintained in non-proprietary Unicode text files.

Modern application software does not maintain the end-user's data in a proprietary workspace format, but instead uses data bases or other data display options, such as Microsoft Office to provide end users the flexibility to analyze their data in their chosen environment. As a .Net language, VisualAPL supports this application system design principle. The Microsoft .Net tools for data access, such as ADO.Net for databases and VSTO for Microsoft Office provide easy to use and superior options for secure and open data storage.

What programming environments are available for VisualAPL?
VisualAPL programmers use the Microsoft Windows operating system environment as their application development environment. VisualAPL provides several development environment options:

- VisualAPL is fully-integrated with the Microsoft Visual Studio IDE, providing 'Intellisense'-incorporated API documentation, a superior debugging and code editing facility and the 'write code and debug/compile cycle' familiar to millions of Windows programmers. The Microsoft Visual Studio IDE is well-suited to creating high-performance, compiled-code applications for multiple deployment scenarios.

  A few of the many benefits of the Visual Studio IDE are:
  - Integrated Source Code Management System
  - Team programming options
  - Enormous code base of 'snippets' as 'how-to-do-it' examples.
  - Common language runtime based on a mal-ware-secure virtual machine.
  - Setup and deployment tools, including web-server-based 1-click installation, are integrated into the IDE.
  - In-line XML code documentation that can be used to produce system documentation, using tools like SandCastle.

- VisualAPL also extends Microsoft Visual Studio with the fully-interactive Cielo Explorer which supports VisualAPL and C# statements. The Cielo Explorer programming IDE is analogous to the statement-by-statement 'session' of traditional APL implementations whereby results of each statement are immediately available for investigation and analysis by the programmer. The Cielo Explorer programming environment is well-suited to experimentation, system design and development. With the availability of the Cielo Explorer,

.Net can be transparently explored and visualized in an immediate-mode session.

- VisualAPL also extends Microsoft Visual Studio with the Cielo Explorer Script Editor. VisualAPL scripts can combine data and functions using VisualAPL or C# statements in a convenient Unicode-based text file. VisualAPL scripts can be executed from any .Net assembly, including those created with C#, VB.Net and VisualAPL. Since VisualAPL scripts are analogous to the 'workspace' of traditional APL implementations, this environment is well-suited to desktop applications used by engineers, scientists, actuaries and other professionals with domain expertise.

- VisualAPL also provides the Lightweight Array Engine in the form of fully-managed .Net assemblies. These assemblies can be referenced by any other .Net language, such as C# or VB.Net, to expose the APL array-based operators directly and without APL special operator characters. The Lightweight Array Engine environment is well-suited to a C# or VB.Net programmer who needs the array-processing and analysis features of VisualAPL using 'keywords' technology rather that the traditional APL 'special character' glyphs.

- Just as with any other .Net language, Microsoft Visual Studio is not required. Programmers may create VisualAPL source code using a Unicode-based text editor, such as Microsoft NotePad and then compile the source code into a .Net assembly using the Microsoft 'command-line' compiler included with the (free) Microsoft .Net SDK. The 'command-line' compiler programming mode is well-suited to the 'spartan' programmer who does not want to license the rich development environment of Microsoft Visual Studio.

If desired, the VisualAPL code developed using any of these programming environments can be used to create mainstream .Net assemblies.

Does VisualAPL use 'special' characters?
Yes, if desired by the programmer. VisualAPL supports the APL 'special' characters that have been traditionally employed as non-intrusive glyphs for APL operators in addition to the familiar add, subtract, multiple, divide operators. These APL operator extensions can simplify the expression of mathematical algorithms and business rules so that program code is less verbose. Traditional APL has generally avoided extensive use of 'keywords'.

Since VisualAPL is Unicode-based, the APL 'special' characters are present in the Unicode character set and implemented in many Unicode-based font sets. Visual Studio provides a convenient tool which a VisualAPL programmer can use to customize the keyboard and character glyph associations to suit their native language.

If desired by the programmer, VisualAPL source code can also contain C# syntax and keywords. This means that code examples from the enormous C# sample code base in the Microsoft Developer Network and other Internet-based sources can generally be used directly in VisualAPL source code.

Does VisualAPL support 32- and 64-bit hardware?
Yes. As a .Net language, the programmer may target the compilation of VisualAPL to either or both hardware environments.

Can VisualAPL implement multi-threaded programming solutions?
Yes, just like any other .Net language. Using Microsoft .Net features for multi-threading, precise control of each thread and thread interaction is easy.

How about exception handling in VisualAPL?
VisualAPL provides the excellent exception (error trapping) facilities of .Net, including try/catch/finally control structure and exception handling of specific error conditions. Because of the interoperability of VisualAPL with other .Net languages, exception handling is transparent among multi-programming-language projects in a .Net application system solution.

VisualAPL also supports the traditional APL quad-error and quad-dm error handling system functions. A VisualAPL extension to quad-dm, quad-dmx provides access to the full error object.

Why does VisualAPL optionally support C# syntax in addition to tradition APL syntax?
By providing C# syntax support directly in VisualAPL:

- The IDE is always up-to-date, providing access to the latest Microsoft .Net developments and enhancements

- Variable scoping is compatible with the mainstream style of C# and other .Net languages.

- Function (method) signatures using traditional APL syntax as well as the extended C# method signatures are available supporting anonymous functions, ‘overloaded’ function definitions and array-based parameter and named function arguments with defaults.

- Extended data types beyond the limited data types of traditional APL are available.

- Like traditional APL implementations, VisualAPL can be used as a dynamically-typed language with bool, string, integer and IEEE double types. With VisualAPL strong data typing is optionally available for enhanced performance and simple interoperability of VisualAPL functions

and mainstream .Net methods. The programmer may choose to use any of the C# strong types to gain easy access to all the functionality of the .Net base classes, for example the DateTime data type and the powerful .Net date arithmetic.

- Besides the 'assignment-by-value' options unique to APL, the mainstream programming standard 'assignment-by-reference' is also available to a VisualAPL programmer

- It is not required to use C# syntax in VisualAPL functions, but its availability in VisualAPL provides a significant advantage when accessing Microsoft .Net features unsupported by traditional APL implementations.

No traditional APL implementation can hope to surpass the universe of Microsoft .Net development tools and documentation. These.Net tools and documentation are always available and up-to-date in VisualAPL because it is a .Net language.

The proprietary duplication of underlying Windows functionality by legacy APL implementations often diluted and delayed the availability of important functionality. Instead, by design, VisualAPL directly exposes all .Net functionality to the interested VisualAPL programmer while still providing the APL syntax and features familiar to a traditional APL programmer.

Can VisualAPL be used to develop web services?
Yes, just like any other .Net language. Traditional 'HTTP(S) Request/Response' as well as WCF (Windows Communication Foundation) web services tools are easy to use from any .Net language, including VisualAPL.

Can VisualAPL be accessed as ActiveX (COM)?
Yes, just like any other .Net language, VisualAPL assemblies can be exposed as COM so that traditional Win32-API-based applications can use them via a .Net 'interop'.

Where can VisualAPL solutions be deployed?
VisualAPL solutions can be deployed in any environment supporting the Microsoft .Net framework.

Just like any other .Net language, VisualAPL solutions can be deployed in the Microsoft Windows environment, the Unix/Linux environment (with the free Microsoft/Novell Mono .Net implementation) and the Mac environment (either natively as a Windows application or using Mono).

Is VisualAPL compatible with Traditional APL syntax?
Yes, VisualAPL provides all the operators and most of the system functions of traditional APL implementations. Some traditional APL system functions cannot

be used in the managed-code environment of .Net, because they could be employed for mal-ware, however .Net alternatives are available, often with superior capabilities.

APL source code has remained largely invariant with respect to operating system platform, and VisualAPL source code is no exception. VisualAPL provides a 'paste APL' edit option which will import legacy APL source code into the Visual Studio environment. A utility for importing legacy APL in 'bulk' is also included with the installation of VisualAPL.

VisualAPL and legacy APL programs (that support an ActiveX interface) can inter-operate easily and efficiently.

Can VisualAPL projects implement an end-user GUI?
Yes. A VisualAPL project can implement an end-user GUI (graphical user interface) in several ways:

- A Microsoft WPF (Windows Presentation Foundation) GUI can be linked to a VisualAPL assembly.
- A Net System.Windows.Forms GUI can be directly constructed from within VisualAPL.
- VisualAPL also provides and extends the traditional APL 'quad-WI' GUI tools of some traditional APL implementations.

What type of application systems can be built with VisualAPL?
VisualAPL has been used to develop mission-critical applications involving mathematically complex actuarial calculations, stochastic modelling, time series analysis and finite-element analysis as well as commercial applications such as high-performance end user form-based access to corporate and government data bases supported by web services and point-of-sale systems for commercial use.

VisualAPL is well-suited to support the calculations and business rules of professional and scientific applications as well as commercial applications, particularly where array-based data analysis is extensively used.

How is VisualAPL maintained?
VisualAPL is maintained by a professional staff with extensive experience with Microsoft .Net languages, APL implementations and application system design and development.

The VisualAPL Forum provides extensive documentation, examples and interaction with other programmers. Because of the unique compatibility of VisualAPL with C# syntax, interaction with mainstream C# programmers is easy and effective.

The VisualAPL development staff has extensive experience in application system design and development and is ready to help you make effective use of the VisualAPL product to implement profitable application systems.

Unlike traditional APL implementations, VisualAPL is based on the Microsoft .Net framework. Thus, VisualAPL does not require development and maintenance efforts associated with proprietary memory management, hardware configuration, APL interpreters, GUI development tools, file system tools, database interface tools, workspace format definitions or data type specifications all of which were required parts of traditional APL implementations. These basic programming 'services' are inherently available in the .Net framework and therefore fully available to the VisualAPL programmer.

How is VisualAPL licensed?
VisualAPL is available in a desktop version for scientists, engineers and other professionals requiring sophisticated programming tools for their own use.

VisualAPL is also available in a professional version for application software developers needing royalty-free distribution rights to distribute their products derived from .Net solutions incorporating VisualAPL.

An enterprise version of VisualAPL is available to provide the professional version of VisualAPL to a large application system development organization at a fixed price.

For qualified education purposes, VisualAPL is available without charge to both instructor and class participants.

For more information about licensing VisualAPL contact sales@apl2000.com.

For more information, a free VisualAPL demonstration/evaluation download and numerous examples and documentation go to the **VisualAPL Forum**: http://forum.apl2000.com/viewforum.php?f=4&sid=55ea912272db8b22d3770b7619a37fda