Graphical User Environments for Scientific Computing

M. Ashworth, R.J. Allan, C.J. Müller, H.J.J. van Dam, W. Smith, D. Hanlon, B.G. Searle and A.G. Sunderland

Computational Science and Engineering Department

CCLRC Daresbury Laboratory, Warrington, WA4 4AD, UK

Abstract

Many scientific applications now have a Graphical User Interface (GUI) customised to make its use more intuitive for novices and experts alike. It is interesting to compare technology designed for GUI development and evaluate the kind of functionality commonly incorporated.

GUIs take on a particular importance in the computational Grid world where aspects of managing applications in a heterogeneous distributed environment can be hidden from most users. Indeed projects like UNICORE provide a GUI specifically designed for submitting HPC applications transparently to a range of different systems. Other similar projects provide seamless access to data across a range of file servers or databases.

Our examples are, in the main, taken from work currently ongoing at Daresbury Laboratory in support of the CCP, HPCI and e-Science programmes. We describe implementations in Java, Perl/Tk, Python/Tk and C++/AVS Express and also C and Perl/CGI. There is particular emphasis on using Web browsers as a special form of GUI because of the current focus on the computational Grid and Internet based distributed services (Web services).

This is a Technical Report of the UKHEC Collaboration.

Report available from http://www.ukhec.ac.uk/publications/reports/guienv.pdf

1 Introduction

1.1 Motivation

Scientific research is relying more and more on the results of computational science, and computer modelling codes are becoming more and more sophisticated. The programs produced by research groups have a greater longevity and there is an increasing need for their use by non-specialists. There is therefore a need to improve the Human Computer Interface (HCI). The interface between the computational kernel of the modelling code and the user should be designed with the following aspects in mind:

- Ease of use
- Automation of data preparation
- Computational steering
- Instantaneous feedback from model runs
- Stepping stone to distributed computing & the Grid
- *Productisation*

The last of these recognises that scientific codes often need to be translated from the scientific world, where the HCI is usually at a rudimentary level, to a commercial product with an interface which is more attractive to a commercial audience.

These aspects have led a number of research groups seriously to embark on the development of Graphical User Interfaces (GUI) for their codes. A GUI is different from a data visualisation system. Hence we have identified both topics as being of importance in enhancing programmers' approaches to computational science on high-end computing platforms. A GUI provides a graphical method of monitoring and control which should ideally be intuitive and hide (and therefore make automatic) complex procedural decisions and operations.

1.2 User Requirements

The user requirements for a particular scientific code may have a number of specialist requirements, but they usually include some of the following:

- Retain old model features - GUI extends capabilities
- Select input data (datasets, files, formats)
- Pre-view & edit input data
- Set run parameters (time step, run length, etc.)
- Select output data (variables, files, formats)

- Run on target system (local or remote, number of processors, etc.)
- Control execution (start, stop, resume, restart etc.)
- Monitor state of model in real-time (select monitor variables)
- Post-view output datasets
- Portable across platforms

1.3 Design considerations

Although it would be optimal for the GUI to be designed as an integral part of the scientific application, it is much more likely that the application code has been developed over many years and that the impulse to add a GUI is much more recent. Therefore the GUI will usually be considered as an add-on to an existing code and there will be little opportunity or motivation to make significant changes to the main code. A very strong constraint on the design of the GUI is, therefore, that it may be integrated with the existing code.

Other considerations include whether the GUI can be distributed as part of a package (e.g. what kind of licence is available), how easy the GUI software is to download and build, and how portable is the resulting GUI/application combination.

1.4 GUIs and the Grid

Probably the most exciting area of applications development on the computational Grid [1] is the construction of Web portals to allow computational scientists, researchers and high performance computer/ application users access to resources via an easy to use Web browser interface. See for example http://dast.nlanr.net/Articles/GridandGlobus/ and http://www.computingportals.org/. The goal is to develop common components that can be used by portal developers to build a Web site that can securely authenticate users to remote resources. Web portals also help users make better decisions for scheduling jobs by allowing them to view pertinent resource information obtained and stored on a remote database or in distributed directories. In addition, profiles are created and stored for portal users allowing them to monitor jobs submitted and view results in a persistent and pervasive way. This is an example of how a GUI can help the user in managing a complex task on a distributed set of resources, without the need to create large files of control parameters or walk through a lengthy command sequence.

1.5 This Document

This paper starts by considering in section 2 some of the technical options available in terms of programming and scripting languages, graphics packages, visualisation and GUI-building toolkits. We also consider some application specific toolkits that have been designed specifically for the computational chemistry area. Sections 3, 4, 5 and 6 examine in detail some applications drawn from computational chemistry, atomic physics and environmental modelling, for which GUIs have been written using a variety of toolkits and environments.

2 Technical Options

2.1 *Programming Languages*

2.1.1 C and C++

Released in 1985, C++ is an object-oriented programming language created by Bjarne Stroustrup. C++ maintains almost all aspects of the C language, while simplifying memory management and adding many features to allow object-oriented programming. C++ is a powerful general-purpose programming language. It can be used to create small programs or large applications.

C and/or C++ may be used to create GUI applications using a wide variety of toolkits, for example:

- Coral (http://www.imonk.com/coraldoc.html)
- FLTK (html://www.fltk.org/)
- FOX (http://www.fox-toolkit.org/fox.html)
- GLOW (http://glow.sourceforge.net/)
- GLUI (http://www.nigels.com/glt/glui/)
- GNOME/GTK+ (http://www.gtk.org/ also available from Perl and Python)
- GNUstep (http://www.gnustep.org/)
- KDE/Qt (http://www.kde.org/)
- Open Motif (http://www.opengroup.org/openmotif/)
- OpenOffice (http://www.openoffice.org/)
- Qt (http://www.trolltech.com/)
- V (http://www.objectcentral.com/vgui/vgui.htm)
- Xmt-Motif Tools (http://motiftools.sourceforge.net/index.html)

to name just a few.

2.1.2 Java

Java is a relatively new language, which has some similarities with C++, but it is much simpler. Many of the more advanced features of object-oriented programming, such as operator overloading, pointer arithmetic and multi-dimensional arrays, were left out. Java source is compiled to *class files*, which contain machine-independent *byte-code*. This is like machine code in form but is not specific to any particular hardware. When a Java enabled client, such as Netscape or Internet Explorer, accesses a page containing a Java applet (a small Java application embedded in a Web page), the byte-code is downloaded and runs on the client's own hardware using an interpreter known as the *Java Virtual Machine*. Because Java is compiled to byte-code, it can run on any platform for which an interpreter has been written. Java is small so it can run efficiently on anything from PCs upwards, and the interpreter takes only a few hundred kilobytes. Java is expanding rapidly and comes complete with a wide range of class libraries and toolkits for Web access, thread based computing and GUIs.

The main advantages of using Java lie in the free availability of Java, a product of Sun Microsystems, its suitability for constructing GUIs of any description, and its universal portability. The result is a portable GUI that can be released freely to all application users and which can be developed as desired by users with special requirements, provided of course they know the Java language which is quite similar, but simpler than, C++.

The standard distribution comes complete with the Abstract Windows Toolkit (AWT) (see section 2.5.1), which provide a set of components that may be used to construct GUIs. A richer, more sophisticated environment is provided by the Java Swing package (see section 2.5.2), which, for recent releases, is now also a part of the Standard Development Toolkit.

See http://java.sun.com/

2.2 Scripting Languages

2.2.1 Tcl and Tcl/Tk

Tk is a widely used graphics library most closely associated with the Tcl (Tool Command Language) language, both of which were developed by John Ousterhout.

Although Tk started out in 1991 as an X11 library, it has since been ported to virtually every popular graphics platform. There are now Tk bindings (e.g. the Tkinter modules described below) for most popular languages, as well as many of the smaller languages.

Tcl is great as a *glue* language. That is, it is extremely easy to add your own commands, written as C functions, and extend the Tcl language with your commands. You can do this in Perl, but it is a lot more difficult. This means that Tcl would be a better choice for embedding an interpreter into your applications especially if you need to call application-specific functions from the embedded interpreter.

Tcl is better than Perl for creating graphical user interfaces. While Perl/Tk makes very good use of the Perl syntax (the Tcl folks could learn a lot from the convenience functions like Scrolled), Perl/Tk still has some problems.

See http://www.scriptics.com/

2.2.2 Perl and Perl/Tk

One of the latest developments in Perl is called Perl/Tk, which combines the Tk toolkit from Tcl with Perl. You use Perl's syntax, creating a Perl script, but create Tk widgets.

This forms a handy way to create easy-to-use interfaces for your Perl scripts. The main problem is that since Perl/Tk is so new, and documentation so scarce, it's hard to get started.

Perl/Tk is an extension to Perl. That means you have to acquire, compile and install Perl/Tk to make any use of this handy extension. Perl/Tk does not come as a part of the standard Perl distribution, except the 5.004_02 version on Windows, which now includes Perl/Tk on Windows for the first time.

When you install Perl/Tk, you do it one of two ways: either you create a shared library that gets dynamically loaded by the perl program, or you create a new Perl interpreter, called tkperl, that includes the Perl/Tk code in the executable. Either method allows you to create user interfaces with Perl/Tk.

If you created tkperl, you can then run the script above with the following command:

```
tkperl tkmenu.pl
```

If you created a dynamically-loadable Perl/Tk library, then you can run the script above with the normal Perl interpreter:

```
perl tkmenu.pl
```

Perl is a great tool to quickly automate tasks, especially tasks involving text files, due to Perl's very strong text-handling and text-searching functions.

Here are some on-line references:

http://www.pconline.com/~erc/perlbook.htm	*Cross-Platform Perl* contains a chapter on Perl/Tk.
http://w4.lns.cornell.edu/~pvhp/ptk/doc/overview.htm	Perl/Tk Overview
http://w4.lns.cornell.edu/~pvhp/ptk/doc/	Tk Docs
http://w4.lns.cornell.edu/~pvhp/ptk/ptkFAQ.html	Perl/Tk FAQ
http://w4.lns.cornell.edu/~pvhp/ptk/ptkPORT.html	Non-Unix Ports

The *CrossThoughts* column in the November 1996 issue of UNIX REVIEW (now UNIX REVIEW's Performance Computing) describes a Perl/Tk menubar.

Steve Lidie writes a Perl/Tk column for the Perl Journal http://www.tpj.com

See http://www.perktk.org/

2.2.3 Python

Python is an interpreted, interactive, object-oriented programming language. It is often compared to Tcl, Perl, Scheme or Java. Python combines remarkable power with very clear syntax. It has modules, classes, exceptions, high-level dynamic data types and dynamic typing. There are interfaces to many system calls and libraries, as well as to various windowing systems, such as X11, Motif, Tk, Mac and Microsoft Foundation Classes (MFC). New built-in modules are easily written in C or C++. Python is usable as an extension language for applications that need a programmable interface.

See http://www.python.org/

2.2.4 Jython

Jython is an implementation of the high-level, dynamic, object-oriented language Python written in 100% Pure Java, and seamlessly integrated with the Java platform. It thus allows you to run Python on any Java platform. Jython is freely available for both commercial and non-commercial use and is distributed with source code. Jython is complementary to Java and is especially suited for the following tasks:

- Embedded scripting - Java programmers can add the Jython libraries to their system to allow end users to write simple or complicated scripts that add functionality to the application.

- Interactive experimentation - Jython provides an interactive interpreter that can be used to interact with Java packages or with running Java applications. This allows programmers to experiment and debug any Java system using Jython.
- Rapid application development - Python programs are typically 2-10x shorter than the equivalent Java programs. This translates directly to increased programmer productivity. The seamless interaction between Python and Java allows developers to freely mix the two languages both during development and in shipping products.

See http://www.jython.org/

2.2.5 BeanShell

BeanShell is a small, free, embeddable, Java source interpreter with object scripting language features, written in Java. BeanShell executes standard Java statements and expressions, in addition to obvious scripting commands and syntax. BeanShell supports scripted objects as simple method closures like those in Perl and JavaScript.

BeanShell can be used interactively for Java experimentation and debugging or as a simple scripting engine for application building. In short: BeanShell is a dynamically interpreted Java, plus some useful stuff. In many ways BeanShell is to Java as Tcl/Tk is to C. BeanShell is embeddable and may be called from Java applications to execute Java code dynamically at run-time or to provide scripting extensibility for applications. Alternatively, BeanShell can call Java applications and objects; working with Java objects and APIs dynamically. Since BeanShell is written in Java and runs in application space, references to Java objects can be freely passed between BeanShell and Java.

See http://www.beanshell.org/

2.3 Graphics Packages

Defined as a library to set up a 2D or 3D scene and to draw low-level objects.

2.3.1 OpenGL

OpenGL is a portable environment for developing interactive 2D and 3D graphics applications. Since its introduction in 1992, OpenGL has become the industry's most widely used and supported 2D and 3D graphics application programming interface (API), bringing thousands of applications to a wide variety of computer platforms. OpenGL contains a broad set of rendering, texture mapping, special effects, and other powerful visualization functions. OpenGL is available across most popular desktop and workstation platforms.

OpenGL is available from Python via the PyOpenGL interface (http://pyopengl.sourceforge.net/).

A Java language interface is available at http://www.jausoft.com/gl4java/

See http://www.opengl.org/

2.3.2 Java3D

Java now provides a wide variety of graphics libraries to support application development. The Java 3D Application Programming Interface (API) is an optional package belonging to a broader set of APIs called the Java Media APIs. Java 3D provides a set of object-oriented interfaces that support a simple, high-level programming model. This enables developers to build, display and manipulate 3D objects and visual environments. Working with Java3D directly is not easy, but it forms the basis of other, higher-level visualisation toolkits (e.g. VisAD).

See http://java.sun.com/products/java-media/3D/

2.4 Visualisation Toolkits

Defined as the drawing of higher-level objects, usually providing some additional support for interactions with scene.

2.4.1 VisAD

VisAD (Visualization for Algorithm Development) is a Java component library for interactive and collaborative visualization and analysis of numerical data. The system combines:

- The use of pure Java for platform independence and to support data sharing and real-time collaboration among geographically distributed users. Support for distributed computing is integrated at the lowest levels of the system using Java RMI distributed objects.
- A general mathematical data model that can be adapted to virtually any numerical data, that supports data sharing among different users, different data sources and different scientific disciplines, and that provides transparent access to data independent of storage format and location (i.e., memory, disk or remote). The data model has been adapted to netCDF, HDF-5, FITS, HDF-EOS, McIDAS, Vis5D, GIF, JPEG, TIFF, QuickTime, ASCII and many other file formats.
- A general display model that supports interactive 3D, data fusion, multiple data views, direct manipulation, collaboration and virtual reality. The display model has been adapted to Java3D and Java2D and used in an ImmersaDesk virtual reality display.
- Data analysis and computation integrated with visualization to support computational steering and other complex interaction modes.
- Support for two distinct communities: developers who create domain- specific systems based on VisAD, and users of those domain-specific systems. VisAD is designed to support a wide variety of user interfaces, ranging from simple data browser applets to complex applications that allow groups of scientists to collaboratively develop data analysis algorithms.

See http://www.ssec.wisc.edu/~billh/visad.html

2.4.2 VTK

The Visualization ToolKit (VTK) is an open source, freely available software system for 3D computer graphics, image processing, and visualization used by a large community of researchers and developers around the world. VTK consists of a C++ class library, and several interpreted interface layers including Tcl/Tk, Java, and Python. Although VTK is freely available, professional support and products for VTK are provided by Kitware, Inc. VTK supports a wide variety of visualization algorithms including scalar, vector, tensor, texture, and volumetric methods; and advanced modelling techniques such as implicit modelling, polygon reduction, mesh smoothing, cutting, contouring, and Delaunay triangulation. In addition, a number of imaging algorithms has been directly integrated to allow the user to mix 2D imaging, 3D graphics algorithms and data. The design and implementation of the library has been strongly influenced by object-oriented principles. VTK has been installed and tested on nearly every Unix-based platform, PCs (Windows 98/ME/NT/2000/XP), and Mac OSX Jaguar or later.

See http://public.kitware.com/VTK/

2.4.3 AVS Express

AVS Express enables object-oriented development of rich and highly interactive scientific and technical data visualizations for Digital, HP, IBM, Linux, SGI, Sun and Windows systems.

Some exercises, which are intended as an introduction to using AVS/Express, and which have a focus on Scientific Visualization are available at http://www.ncsc.org/training/materials/express_workbook/

See also http://www.avs.com/software/soft_t/avsxps.html

2.5 GUI Widget Toolkits

2.5.1 Java AWT

The AWT (Abstract Window Toolkit) is part of the Java standard release and is designed for creating user interfaces and for painting graphics and images. It is a set of classes intended to provide everything a developer requires in order to create a graphical interface for any Java applet or application. Most AWT components are derived from the java.awt.Component class. AWT features include:

- a rich set of user interface components;
- a robust event-handling model;
- graphics and imaging tools, including shape, colour, and font classes;
- layout managers, for flexible window layouts that don't depend on a particular window size or screen resolution;
- data transfer classes, for cut-and-paste through the native platform clipboard.

2.5.2 Java Swing

The Swing package is part of the Java Foundation Classes (JFC) in the Java platform. The JFC encompasses a group of features to help people build GUIs and Swing provides all the components from buttons to split panes and tables. "Swing" was the code name of the project that developed the new components. Although it is an unofficial name, it is still frequently used to refer to the new components and related API and is immortalised in the package names. Swing is a large set of components ranging from the very simple, such as labels, to the very complex, such as tables, trees, and styled text documents. Almost all Swing components are derived from a single parent called JComponent that extends the AWT Container class. Thus, Swing is best described as a layer on top of AWT rather than a replacement for it.

The Swing package was first available as an add-on to JDK 1.1. Prior to the introduction of the Swing package, the Abstract Window Toolkit (AWT) components provided all the UI components in the JDK 1.0 and 1.1 platforms. Although the Java 2 Platform still supports the AWT components, most experts strongly encourage the use of Swing components. Swing components are recognisable because their names start with "J". For example, the AWT class to implement a checkbox is named Checkbox, whereas the Swing checkbox class is named JCheckBox. In addition, the AWT components are in the java.awt package, whereas the Swing components are in the javax.swing package.

As a rule, programmers should not use "heavyweight" AWT components alongside Swing components. Heavyweight components include all the ready-to-use AWT components, such as Menu and ScrollPane, and all components that inherit from the AWT Canvas and Panel classes. When Swing components (and all other "lightweight" components) overlap with heavyweight components, the heavyweight component is always painted on top.

The Swing Component Set provides a set of "lightweight" (all-Java language) components that, to the maximum degree possible, work the same on all platforms. Swing provides many standard GUI components such as buttons, lists, menus, and text areas, which you combine to create your program's GUI. It also includes containers such as windows and tool bars. On the downside, you may find that Swing applications are slower and more memory hungry than their AWT counterparts.

See http://java.sun.com/docs/books/tutorial/uiswing/

2.5.3 Tcl/Tk

Tcl, (Tool Command Language), pronounced *tickle*, is an interpreted language with programming features, available across platforms running Unix, Windows and the Macintosh operating system. Tk, the associated toolkit is an easy and efficient way of developing window-based applications. Application tasks may be split into modules and any new application specific task may be written and compiled as a C or C++ program and exported as a new Tcl command. Then a Tcl script is composed to drive the overall application. The scripting language, much like any shell language, has the ability to access and execute any other programs. Therefore several Tcl based applications could be made to work together to create or extend into a new application.

Tcl consists of a few syntax rules and a (still growing) set of core commands. Tk provides a higher level application programming interface for developing interactive widget-based applications, particularly for those who wish to concentrate on their application and who have no need to gain in-depth programming expertise in the underlying window system or verbose toolkits such as OSF/Motif. Tcl/Tk is free, available now on Unix, Windows and Macintosh, and has a wide user base with a rich and growing mass of useful contributed software. The wide availability of and the ease of teaching and learning Tcl/Tk makes it a popular tool for teaching the principles of Graphical User Interface design and development.

Tk has been created to give the Tcl language a graphical toolkit. It is a multi-platform graphical toolkit with the look of the native OS. The great advantage of Tcl/Tk is its simplicity. There is also a full-featured GUI builder for Tk called xf which allows you to build live interfaces without writing any code.

See http://www.tcl.tk/software/tcltk/

2.5.4 Python Tkinter

Tkinter is Python's de-facto standard GUI (Graphical User Interface) package and is part of the standard Python distribution [16]. It is a thin object-oriented layer on top of Tcl/Tk. To use Tkinter, you do not need to write Tcl code, but you will need to consult the Tk documentation and occasionally the Tcl documentation (since Tk's low-level event handling mechanism is considered part of Tcl).

Tkinter is not the only GUI for Python. It is however the most commonly used one, and almost the only one that is portable between Unix, Mac and Windows. The Tkinter module ("Tk interface") is the standard Python interface to the Tk GUI toolkit from Scriptics (http://www.scriptics.com/).

Both Tk and Tkinter are available on most Unix platforms, as well as on Windows and Macintosh systems. Starting with the 8.0 release, Tk offers native look and feel on all platforms. Tkinter consists of a number of modules. The Tk interface is located in a binary module named _tkinter (this was tkinter in earlier versions). This module contains the low-level interface to Tk, and should never be used directly by application programmers. It is usually a shared library (or DLL), but might in some cases be statically linked with the Python interpreter.

See http://www.python.org/topics/tkinter/

2.5.5 Python wx

A GUI library for Python based on wxWindows: a blending of the wxWindows C++ class library with the Python programming language. wxPython is a GUI toolkit for the Python programming language. It allows Python programmers to create programs with a robust, highly functional graphical user interface, simply and easily. It is implemented as a Python extension module (native code) that wraps the popular wxWindows cross platform GUI library, which is written in C++.

Like Python and wxWindows, wxPython is open source that means that it is free for anyone to use and the source code is available for anyone to look at and modify. Developers can also contribute fixes or enhancements to the project. wxPython is a cross-platform toolkit. This means that the same program will run on multiple platforms without modification. Currently supported platforms are 32-bit Microsoft Windows, most Unix or Unix-like systems, and Macintosh OS. Since the language is Python, wxPython programs are simple, easy to write and easy to understand.

See http://www.wxpython.org/

2.6 Distributed Computing

Adding a Graphical User Interface to an application allows the user to feel much closer to the computation and raises the question of whether the user can see results from the simulation in real-time and whether he/she can steer the application while it is running. This potentially links GUIs with the large and developing area of distributed computing, Grid computing, meta-computing and computational steering.

We therefore describe here a few of the common tools used in this area, but a full survey is available [2].

2.6.1 CORBA

The Common Object Request Broker Architecture (CORBA) (Mowbray and Zahari, 1995, Siegel, 2000) is a low-level architecture established in 1989 by the Object Management Group (OMG) (http://www.omg.org/). CORBA is an open, vendor-independent architecture and infrastructure that applications can use to work together over networks. Using the standard Internet Inter-ORB Protocol (IIOP), built on top of TCP/IP, a CORBA-based program from any vendor, on almost any computer, operating system, programming language, and network, can inter-operate with any other CORBA-based program.

CORBA provides a specification for the Interface Definition Language (IDL). IDL lets developers define interfaces to their programs and objects in a standardized fashion. With the IDL are mappings that map the IDL definitions and types to programming languages such as C, C++ and Java. CORBA offers developers complete language transparency.

Developer and vendor objects interact with one another through an Object Request Broker (ORB). Using the language mappings, developers can create client-side "stubs" and server-side "skeletons" that their ORBs will understand.

CORBA applications are composed of objects. For each object type you define an interface in IDL. The IDL interface defines the syntax for the contract that the server object offers to the clients that invoke it. Any client that wants to invoke an operation on the object *must* use this IDL interface to specify the operation it wants to perform, and to marshal the arguments that it sends. When the invocation reaches the target object, the *same* interface definition is used there to unmarshal the arguments so that the object can perform the requested operation with them. The interface definition is then used to marshal the results for their trip back, and to unmarshal them when they reach their destination.

The IDL interface definition is independent of programming language, but maps to all of the popular programming languages via OMG standards. OMG has defined standard mappings from IDL to C, C++, Java (Brose et al, 2001), COBOL, Smalltalk, Ada, Lisp, Python, and IDLscript.

The OMG provide a specification – not an implementation. It is up to other individuals, groups and companies to provide implementations.

Fnorb is a CORBA 2.0 ORB for Python, see http://www.fnorb.org

See also http://www.omg.org/

2.6.2 Java RMI

Java Remote Method Invocation (RMI) enables the Java programmer to create distributed Java-to-Java applications, in which the methods of remote Java objects can be invoked from other Java virtual machines, possibly on different hosts.

A Java program can make a call on a remote object once it obtains a reference to the remote object, either by looking up the remote object in the bootstrap naming service provided by RMI or by receiving the reference as an argument or a return value. A client can call a remote object in a server, and that server can also be a client of other remote objects. RMI uses object serialisation to marshal and unmarshal parameters and does not truncate types, supporting true object-oriented polymorphism.

RMI applications often use the client-server model. A typical server application creates a number of objects, makes references to those objects accessible remotely and waits for clients to invoke methods on those remote objects. A typical client application gets a remote reference to one or more remote objects on the server and then invokes methods on them. RMI provides the mechanism by which the server and the client communicate and pass information back and forth. Such an application is sometimes referred to as a distributed object application.

See http://java.sun.com/products/jdk/rmi/index.html

2.6.3 Globus

The Globus project is developing basic software infrastructure for computations that integrate geographically distributed computational and information resources.

GLOBUS is a joint project of Argonne National Laboratory, the University of Chicago and the University of Southern California's Information Sciences Institute as well as a large number of other collaborators.

Core components include:

- GRAM — Globus Resource Allocation and process Manager provides uniform resource allocation, object creation, computation management and co-allocation mechanisms for diverse resource types;
- Nexus — heterogeneous communication infrastructure, supports uni-cast and multi-cast;
- MDS — Monitoring and Discovery Service, structure and state information;
- GSI — Grid Security Infrastructure, authentication and related security services, provides public key based single sign-on, run-anywhere capabilities for multi site environments. GSI supports proxy credentials, inter-operability with local security mechanisms, local control over access, and delegation. A wide range of GSI-based applications has been developed ranging from ssh and ftp to MPI, Condor and the SDSC Storage Resource Broker;
- GASS and GEM — Global Access to Secondary Storage and Global Executable Manager. GASS provides a uniform name space (via URLs) and access mechanisms for files accessed via different protocols and stored in diverse storage system types (HTTP, FTP, HPSS, DPSS etc.).

Participants in the GUSTO Consortium of NPACI sites are testing GLOBUS concepts on a global scale.

GLOBUS is also the first software on the TransPAC network which links the Asia Pacific Advanced Network (APAN) and vNBS academic research networks, see http://www.transpac.org/. Many scientific applications have now been ported.

Several steps are required in writing an application to use GLOBUS. Each component of the GLOBUS toolkit may be used independently or in connection with the other services. GRAM can be used for process creation and control. It acts as a "gatekeeper" that first creates certificates of authenticity for each user enabling access to remote compute resources. It interfaces to local resource management software such as NQE or LSF. Once processes are executing, GRAM provides the user with a means to check job status, kill jobs or read output. GASS can be used to access common persistent storage. Servers essentially allow remote processes, which use the GASS client utilities or library functions, to access local file systems. GASS servers thus act as simple file servers binding to a port and transferring files to and from the local file system driven by requests from remote processes. The MDS is based on the Lightweight Directory Access Protocol (LDAP). This acts as a general-purpose repository for information about resources in the GLOBUS testbed. Among other data it stores information about where each gatekeeper is running, how to contact it (i.e. TCP/IP port number) and how many nodes are free on the resource it manages.

See http://www.globus.org/

2.6.4 OGSA

The Open Grid Services Architecture (OGSA) is a proposed evolution of the current Globus Toolkit towards a Grid system architecture based on an integration of Grid and Web services concepts and technologies. Initial proposed technical specifications have been developed by the Globus Project and IBM, and are being put forward at the Global Grid Forum for discussion, refinement, and eventual standardisation.

The Globus Project is committed to developing a high-quality open source OGSA implementation. During 2003 and into 2004, the current Globus Toolkit will evolve towards an OGSA-compliant Globus Toolkit, in a manner that will address backward compatibility and transitioning issues.

See http://www.globus.org/ogsa/

2.7 Application Specific

2.7.1 PyMOL: A Python based molecular graphics system

PyMOL [9] developed by W. DeLano is an Open Source molecular graphics system. At the heart of the application lies the Python interpreter. Python [21] is an interpreted object oriented language. It supports modules, classes, exceptions, high-level dynamic data types, and dynamic typing (i.e. no need to specify variable types). Libraries written in C, C++ or Fortran can extend the basic functionality. A wide variety of such libraries are currently available, offering access to operating system functions (like sockets, directory structures, and such) or application-specific capabilities (e.g. graphics, numerical mathematics or molecular mechanics). To access the functionality of such libraries they only have to be copied to an appropriate directory upon which they can be imported with a standard Python command. No recompilation or reconfiguration of the interpreter is needed making this approach particularly flexible. This means that extensions can be distributed without the requirement that the user has to have access to a specially modified version of the interpreter. In addition to this there are many pure Python libraries available as well. Furthermore, Python offers automated memory management through run-time reference counting and garbage collection of unreferenced objects.

Apart from the basic Python language the Python environment comes with variety of extension and libraries. The most important one for this current article is Tkinter [16]. Tkinter is a Python binding of the Tk [19] widget set. The Tk widgets offer quite elementary functions though and therefore are not always an ideal solution for recurring types of more complex functions. This issue has been addressed by the Python Megawidget (Pmw) library [17], which builds on Tkinter to provide higher level widgets like menubars, notebooks, and various dialog boxes. Thus an environment which offers powerful building blocks for GUI development is available.

PyMOL extends the basic Python environment in at least three ways. First, it provides an OpenGL based renderer for drawing chemical objects. This C-implemented renderer is designed to be as thin and efficient as possible. The renderer has Python bindings which offer a powerful API allowing editing of chemical structures with different options including ribbon representations of biomolecular molecules, and the generation of molecular dynamics movies.

Second, the Chempy package provides a native Python representation of chemical structures. It provides definitions for chemical building blocks from atoms to general molecules, as well as the capability to read and write a variety of file formats commonly used in chemistry. In addition to this a variety of editing operations on the chemical structures are implemented. Thus Chempy provides Python with the chemistry machinery required.

Third, it offers a Tkinter [16] based GUI. The user interface provides access to a variety of options and operations, including a "command line" from which commands can be entered interactively. The GUI also provides balloon help and online help functions, although the online help currently consists of plain text being dumped to a text widget. Because the GUI is build up like a framework using an object oriented approach it can be modified in powerful ways to specialise PyMOL for specific application areas or to integrate it into other applications.

The current version of PyMOL offers a free application supporting many functions useful to the chemist, like real-time 3D structure visualisation, various molecular representations including ribbons, lines and spheres, input and output of Protein Databank (PDB) and Macromodel files, as well as image output using libPNG.

See http://www.pymol.org/

2.7.2 NanoCAD

NanoCAD is an applet driving 2D graphics, written in Java by Will Ware at MIT. The source code is available from the NanoCAD web site (see below). NanoCAD was largely inspired by the book *Unbounding the Future*, in which the second chapter describes a virtual-reality simulation of the molecular-scale world.

NanoCAD uses the mathematical techniques of molecular modelling to simulate the behaviour of molecules on your web browser. An "Energy minimize" button performs energy minimization, i.e. NanoCAD tries to nudge the molecule toward a shape with a lower potential energy. Real molecules jiggle around their minimal-energy shapes due to thermal vibrations.

Previous versions of NanoCAD were written in either Scheme or Common Lisp. This proved inconvenient for many people, and the new Java version should make NanoCAD more widely available. NanoCAD is free software, available under license.

The most complete, complex and computationally intensive chemistry simulations are *ab initio* calculations, which find solutions to Schrodinger's wave equation, solving for the exact shapes of electron clouds. In such simulations, chemical bonds are emergent phenomena and are not a-priori built into the simulation. By contrast, semi-empirical modelling is used for systems that depend partially on empirically gathered data, and molecular mechanics depends entirely on empirical data to make up mass-and-spring pseudo-mechanical models of molecular behaviour.

NanoCAD is based on molecular mechanics. The mathematical model of molecular mechanics used by NanoCAD, called MM2, was developed by Norman Allinger of the University of Georgia. MM2 is described in Eric Drexler's book "Nanosystems", starting on page 44. MM2 essentially treats a group of atoms as a collection of masses, non-linear springs, torsion bars, and so on.

See http://willware.net:8080/ncad.html

2.7.3 Jumbo 3

Java Universal Markup Browser for Objects (Jumbo) is a Java-based molecular browser and editor for molecular data files stored primarily as Chemical Markup Language (CML) and XML documents. CML is a new approach to managing molecular information using recently developed Internet tools such as SGML/XML and Java. It is based strictly on SGML, the most robust and widely used system for precise information management in many areas. CML is capable of holding extremely complex information structures and so acting as an interchange mechanism or for archival. It interfaces easily with modern database architectures such as relational databases or object-oriented databases.

See http://www.xml-cml.org/jumbo3/

2.7.4 MMTK

The Molecular Modelling Toolkit (MMTK) is an Open Source program library for molecular simulation applications. In addition to providing ready-to-use implementations of standard algorithms, MMTK serves as a code basis that can be easily extended and modified to deal with standard and non-standard problems in molecular simulations.

MMTK is developed in and around Python, a high-level object-oriented general-purpose programming language. In fact, MMTK consists of nothing more than a collection of Python modules, most of which are written in Python itself, with only a small time-critical part (e.g. energy evaluation) written in C. MMTK applications are Python programs that make use of these modules. Python was chosen because it allows rapid code development and testing, while providing a very convenient C interface for dealing with time-critical calculations.

MMTK is based on an object-oriented model of molecular systems. A system is made up of atoms, molecules, and complexes, all of which are defined in MMTK's chemical database. A molecule, for example, is defined in terms of atoms, functional groups, bonds, force field parameters, etc. It is possible to introduce specialised versions of these objects; for example, MMTK has special support for proteins, which are basically chemical complexes, but can be handled in terms of peptide chains, residues, side-chains etc.

MMTK's capabilities include:

- construction of molecular systems, with special support for proteins and nucleic acids;
- infinite systems or periodic boundary conditions (orthorhombic elementary cells);
- common geometrical operations on co-ordinates;
- rigid-body fits;
- visualization using external PDB and VRML viewers; animation of dynamics trajectories and normal modes;
- the AMBER 94 force field, with several options for handling electrostatic interactions;
- a deformation force field for fast normal mode calculations on proteins;
- energy minimisation (steepest descent and conjugate gradient);
- molecular dynamics (with optional thermostat, barostat, and distance constraints);
- normal mode analysis;
- trajectory operations;
- point charge fits;
- molecular surface calculations; and
- interfaces to other programs.

Compared to standard modelling code written in Fortran, MMTK is much easier to understand, extend, and modify. For example, new force fields can be added without touching any existing code, i.e. without any risk of breaking it, and new integrators can be developed without any assumptions about force field implementations. MMTK users can also profit from a large collection of Python code developed for other applications, scientific or otherwise.

See http://starship.python.net/crew/hinsen/MMTK/

2.7.5 PMV

The Python Molecule Viewer (PMV), from the Scripps Research Institute, has been developed on top of the MolKit, DejaVu and ViewerFramework packages. PMV is a general purpose viewer that can be integrated into any computational chemistry package available in Python. It relies on DejaVu for the 3-Dimensional visualization, ViewerFramework for the definition of individual commands mglutil for the GUI and MolKit for the representation of molecules.

This viewer has most of the features usually expected in a molecule viewer:

- stick and sphere (Corey-Pauling-Koltun or CPK) representation;
- different colouring schemes (by atom, by residue type, by chain, by molecule, by properties, etc...);
- measuring tools;
- atom identification by picking;
- support for multiple molecules;
- secondary structure representation; and
- user definable sets of atoms, residues, chains and molecules etc.

In addition to these traditional features it is dynamically extensible, i.e. new commands can be developed independently and placed in libraries. The Viewer inherits from the ViewerFramework the capability dynamically to import these commands as needed. In fact, all commands in the viewer have been developed based on this principle. This provides a way to add features to the application that is incremental and well suited for team development. In addition this approach avoids the "feature overload" problem, i.e. overloaded menus cluttered with commands that are irrelevant for the problem at hands. Customisation files allow users, among other things, to specify which commands should be loaded when the application starts. This allows users to define a number of molecular viewing applications just by creating different customisation files.

See http://www.scripps.edu/~sanner/python/documentation/index.html

2.8 HTML, Perl/CGI and C/CGIC

Perl has made great strides as the language of choice for Common Gateway Interface, or CGI scripts. CGI scripts act as the gateway between the Web and outside sources of data, such as databases or online catalogues.

A CGI script runs on the Web server machine (not in your Web browser) and dynamically creates Web pages. CGI scripts also respond to data you enter into Web forms.

Due to its strong text-handling capabilities, Perl works very well as a CGI script language. In addition, as a scripting language, Perl provides easy facilities to launch other applications and display the results, such as displaying the results of a database query in a Web page.

In Perl, the easiest way to start writing CGI scripts is through the CGI.pm Perl module for working with CGI scripts. This module contains Perl subroutines to output HTML data, such as the HTML header block.

Here are some references:

Chapter 9 of *Cross-Platform Perl*

http://www.pconline.com/~erc/perlbook.htm

http://www.mispress.com/introcgi/ An Introduction to CGI/Perl http://agora.leeds.ac.uk/Perl/Cgi/start.html CGI Tutorial: Start

http://hoohoo.ncsa.uiuc.edu/cgi/interface.html The Common Gateway Interface Specification

http://hoohoo.ncsa.uiuc.edu/cgi/primer.html The Common Gateway Interface Primer (NCSA)

3 CCP1 PyMOL GUI development for molecular quantum chemistry

by Huub van Dam

The Quantum Chemistry Group at Daresbury Laboratory that supports CCP1 was confronted with a double-headed problem. Firstly we were faced with a long-standing requirement for a graphical user interface for our main *ab-initio* code GAMESS-UK. We see this as an important issue as a good GUI can enhance productivity considerably, flatten the learning curve, and avoid human input errors. Secondly, the UK electronic structure theory community (CCP1) required a free, extensible GUI that could be rapidly customised for all its codes. This GUI was to be used for teaching purposes, among other things.

Previous projects had taught us a number of lessons.

- Libraries like X11 and Motif are very versatile but extremely tedious to use. Also portability is an issue as there still is no standard window library across the Unix, Windows and MacOS platforms.
- Scripting languages offer powerful ways to combine various programs to do something new. Also they are very useful for prototyping because of their interpreted nature and fault tolerance.
- The Tk set of widgets offer a very portable and extensive set of graphical elements.
- Object Oriented programming is very useful for GUI development.
- The OpenGL library offers very powerful and portable 3D graphics capabilities.

Together with the requirements of our community that the solution should be free and extensible PyMOL seems a suitable answer.

Extending PyMOL

The original GUI provided with PyMOL was built in a framework fashion using an application base class. From this starting point we chose to proceed by defining the CCP1 GUI to be a sub-class of PyMOL. This sub-class specialises PyMOL by adding a new menu that gives access to *ab initio* functionality that we provide.

The actual *ab initio* functionality is organised in "calculations". A calculation is defined as an object that holds all input values, model parameters and results of some computational process. As such a calculation is a very general concept. For our purposes we have defined a base class to implement this concept. From this class we derived an *ab initio* calculation class, which in turn forms the basis for program specific calculation classes.

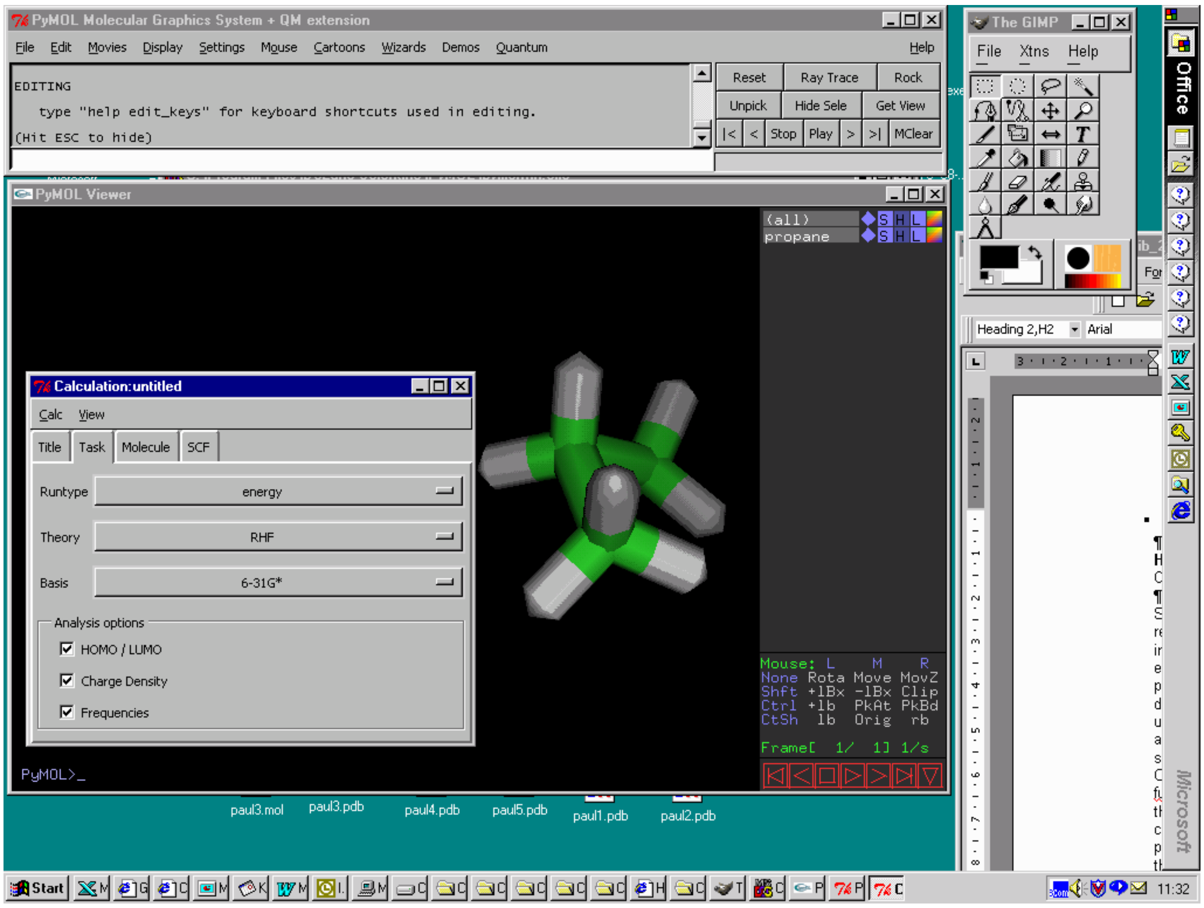
The calculation classes themselves actually work as storage classes. Calculation editors perform the graphical representation and manipulation of their contents. The hierarchy of the calculation editors mirrors that of the calculation classes. The main advantage of this is that the contents of the calculation object are completely independent of the runtime state of the GUI. This enables using Pythons Pickle module for saving and loading calculation objects, which saves writing special I/O routines. Furthermore it becomes possible to write completely different GUIs to interact with the same calculation storage classes if that were ever needed. In addition to this we have limited the number of references to the PyMOL renderer. This means that we could even replace the current renderer with another one without too much work.

The current calculation editors are based on notebooks as provided by the Pmw library. Different tabcards in the notebook deal with different aspects of the calculation. In practice the base calculation editor class provides the notebook and each sub-class adds tabcards appropriate for the level of abstraction it addresses. This approach supports code reuse, while at the same time organising the calculation parameters in a way convenient to the user.

Although most data items in a calculation can be handled very well within the context of a notebook based calculation editor some objects require special attention. In particular the editing operations for molecular structures within PyMOL itself are rudimentary. Therefore a good quality molecule editor is much needed and serves an important purpose beyond our particular needs. Also the Z-matrix format used in the quantum chemistry community requires some special attention. For these reasons a molecule editor is developed as a separate component. It is capable of editing a molecule in a Cartesian co-ordinate representation as well as in a Z-matrix representation. A particularly powerful capability is that the user can at any time switch from one representation to the other and can continue editing. All conversion related issues are handled automatically.

An advanced issue resulting from the calculation concept is that after a calculation has started something needs to keep track of its progress and retrieve results, as they become available. This is particularly important as *ab initio* calculations can take several days to complete so that it is unreasonable to expect the user to wait while it is running. To address this issue we have implemented a job-manager class. A job-manager instance takes the instructions for a calculation and spawns a separate thread to run it. This releases the GUI so that the user can view results as and when they become available. The current job-manager has rather limited capabilities. However it could be extended to become grid aware using the pyGlobus [15] package.

Finally to demonstrate the extensibility of the solution we have chosen, we have build in support for two *ab initio* programs using the same hierarchy of classes and concepts. Currently the GUI supports both GAMESS-UK and CADPAC. In future work the GUI will be extended to support a wide range of ab-initio codes.


PyMOL Molecular Graphics System + QM extension
File Edit Movies Display Settings Mouse Cartoons Wizards Demos Quantum Help
Reset Ray Trace Rock
Unpick Hide Sele Get View
|< < Stop Play > >| MClear
PyMOL Viewer
Calculation:untitled
Calc View
Title Task Molecule SCF
The GIMP
File Xtns Help
Heading 2,H2 Arial
Office
paul3.mol paul3.pdb paul4.pdb paul5.pdb paul1.pdb paul2.pdb
Start
11:32
Microsoft

```
EDITING
   type "help edit_keys" for keyboard shortcuts used in editing.
(Hit ESC to hide)
```

```
(all)      S H L
propane    S H L
```

Runtype: energy

Theory: RHF

Basis: 6-31G*

Analysis options

- [x] HOMO / LUMO
- [x] Charge Density
- [x] Frequencies

```
PyMOL>_
```

```
Mouse: L    M    R
None Rota Move MovZ
Shft +lBx -lBx Clip
Ctrl +lb  PkAt PkBd
CtSh  lb  Orig  rb
Frame[  1/  1] 1/s
```

Figure 1. A screen shot of the PyMOL molecular graphics system showing the notebook widget used to set the quantum chemistry options.

In summary, the PyMOL program building on Python, OpenGL, and Tk offers a portable and extensible environment for molecular chemistry GUI development (Figure 1). In particular, the object oriented nature of Python, as well as its capability to import new functions from shared libraries, make Python a powerful and flexible tool.

4 Development of a configuration GUI for the CCP2 PFARM code

By Andrew Sunderland

Perl/Tk was chosen to develop a configuration interface for a Fortran application in atomic scattering theory, which needed an initial data file specifying the task to be carried out. Perl was found to be very well suited to the small calculations required to determine the configuration from the user's input and this is combined with the ease and flexibility of the Tk widgets for producing and controlling graphical objects.

The basic layout of the first GUI for this application was coded and run in under five minutes, see Figure 2.

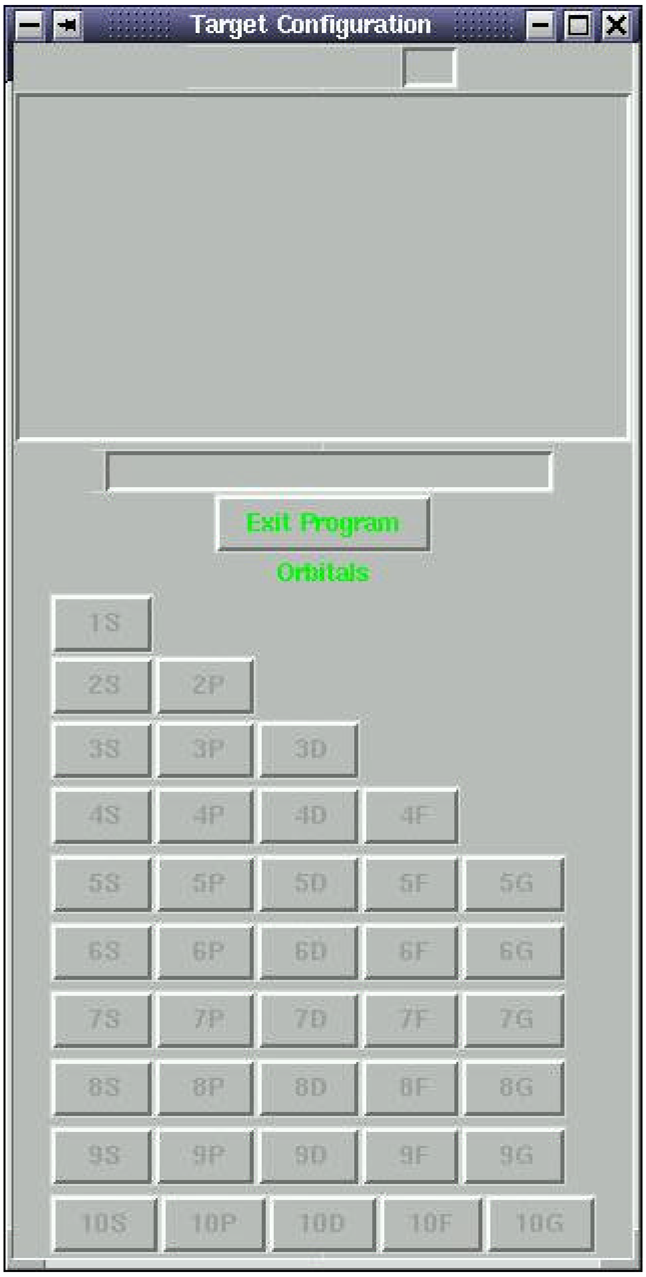


Figure 2. The basic layout of the first GUI prototype for the CCP2 atomic scattering application

The object-oriented Perl/Tk environment sets up this main window as follows:

```
# create widget object
    my $WIDGET = MainWindow->new;
# set title of this object
    $WIDGET->title("Target Configuration");
# set background colour
    $WIDGET->configure(-bg=> 'gray', -width=> 57);
```

Frames and sub-frames were used quickly to create the desired layout:

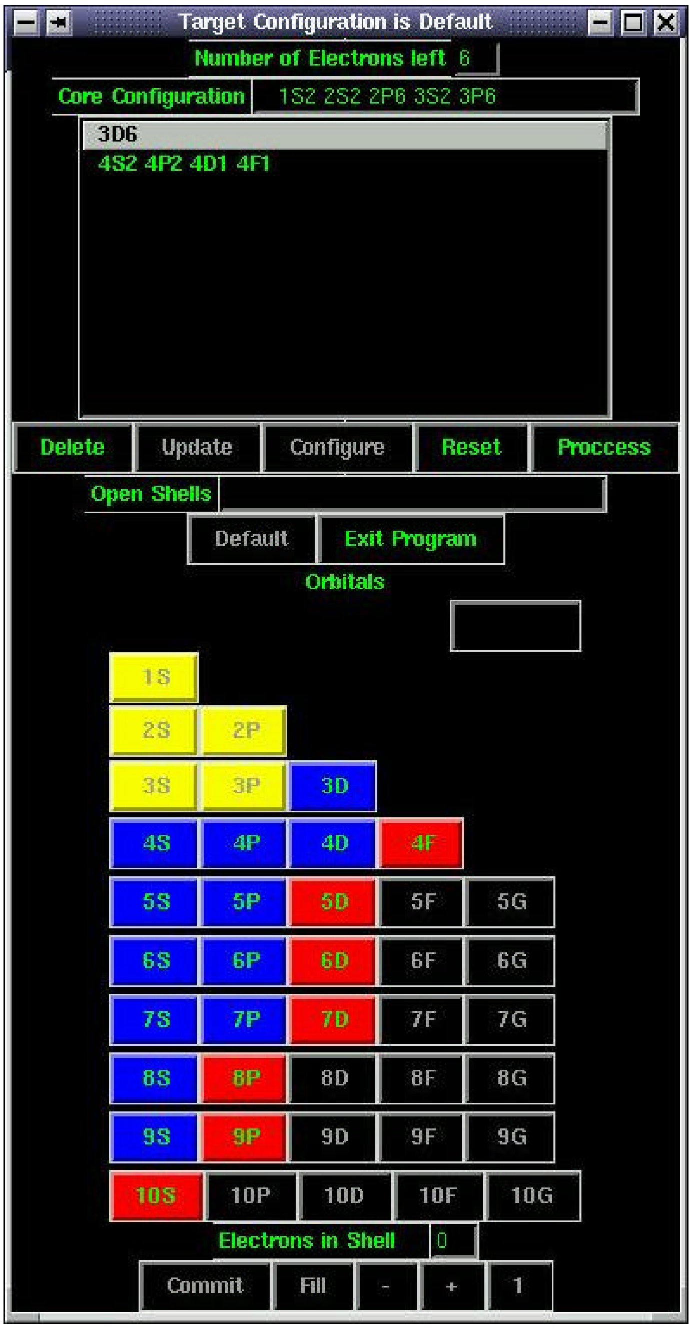
```
    for($b = 0; $b <= 11; ++$b) {
         $FRAME[$b] = $WIDGET->Frame->pack(-side=> 'top');
    }
    $FRAME_O = $FRAME[9]->Frame->pack(-side=> 'left');
    $FRAME_O->configure(-bg=> 'gray');
    $LABEL_O  = $FRAME_O->Label(-text=> "Orbitals", -fg=>
'green', -bg=> 'gray')->pack(-side=> 'top');
    $RANGE = @N_LIST;
    for($b = 0; $b < $RANGE; ++$b) {
        $NUM = @N_LIST[$b];
        $FRAME_N[$NUM] = $FRAME_O->Frame->pack(
                   -side=> 'top', -anchor=> 'w');
        $FRAME_N[$NUM]->configure(-bg=> 'gray');
    }
```

Buttons may then be added for the required functionality of the final widget. It is important to include an “exit" button:

```
    $BUTTON_EXIT = $FRAME[8]->Button(
        -text=> "Exit Program", -command=> sub{exit;},
        -fg=> 'green', -bg=> 'gray')->pack(-side=> 'right');

    for($b = 0; $b < $RANGE; ++$b) {
        if(@N_LIST[$b] < 10) {
     $SHELL = " @N_LIST[$b]@L_LIST[$b]";
        }
        else {
     $SHELL = "@N_LIST[$b]@L_LIST[$b]";
        }
        $NUM = @N_LIST[$b];
        $SHELL_NUM = $b;
        $BUTTON[$b] = $FRAME_N[$NUM]->Button(-text=> $SHELL,
     -fg=> 'green', -bg=> 'gray',
     -command=> sub{CONFIGURATIONS($Value = $b);},
            -state=> 'disabled')->pack(-side=> 'left', -
anchor=> 'w');
```

A listbox was added to the widget, in which configurations could be shown. It was after this that “Label" and “Entry" widgets could be added:

```
    $LB1 = $FRAME[3]->Scrolled(
        "Listbox", -scrollbars=> 'oe', -fg=> 'green',
        -bg=> 'gray', -width=> 40)->pack(
        -side=> 'left', -fill=> 'both');
    $LB1->configure(-selectmode=> 'single');

    $FRAME[1]->Label(-textvariable=> $TEXT, -fg=> 'green',
      -bg=> 'gray')->pack(-side=> 'left');
    $FRAME[1]->Entry(-textvariable=> $EVAL, -fg=> 'green',
      -bg=> 'gray', -width=> 3, -state=> 'disabled')->pack(
      -side=> 'left');

    $FRAME[7]->Label(-textvariable=> $MODE_TEXT, -fg=>
'green',
      -bg=> 'gray')->pack(-side=> 'left');
    $FRAME[7]->Entry(-textvariable=> $TEMP_CONF, -fg=>
'green', -bg=> 'gray', -width=> 31, -state=> 'disabled')-
>pack(-side=> 'left');
```


Target Configuration is Default

Number of Electrons left 6

Core Configuration 1S2 2S2 2P6 3S2 3P6

3D6
4S2 4P2 4D1 4F1

Electrons in Shell 0

Figure 3. The final GUI layout for the CCP2 atomic scattering application

To make rapid progress it is useful to have a reasonable amount of knowledge about windows programming environments in order to understand what the widgets do. As the GUI started to take shape, so procedures could be added to give it additional capabilities. In less than 20 hours a working GUI application was produced, with all the basic required functionality as shown in Figure 3.

Such quick and easy development is a great promotion for Perl/Tk. Perl is an interpreted script language which makes it very quick and easy to make modifications. This is equally important for the development of Web-based GUIs using Perl/CGI. The code can also be compiled in to an executable.

5 The POLCOMS Java GUI

by Mike Ashworth

5.1.1 Background

The Proudman Oceanographic Laboratory (POL) has a world-class program of simulation of shelf seas using computational hydrodynamic models. Such computer models have a wide range of applications, including coastal engineering, offshore industries, fisheries management, marine pollution monitoring, weather forecasting and climate research.

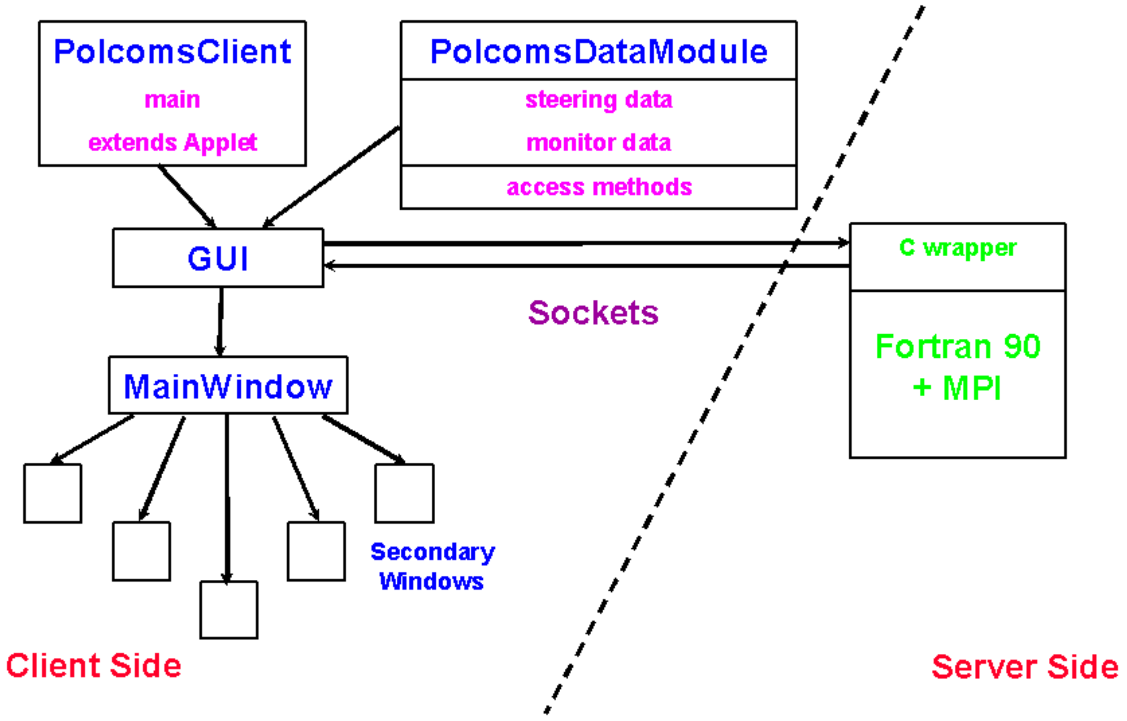


Figure 4. Structure of the POLCOMS Java GUI

CCLRC Daresbury Laboratory is collaborating with POL on the development of a three-dimensional model, which covers the northwest European shelf, and extending to beyond the shelf edge. The hydrodynamic model can be coupled to an ecosystem model in order to simulate the development of plankton blooms and to study the effects of anthroprogenic influences on the marine ecosystem.

5.1.2 Technology Decisions

Java is a modern, object-oriented programming language that has been designed from the start to facilitate the provision of services over the Internet. Java offers many advantages including robustness, ease of programming and reusability, but the key features for this project are its portability and the access to a rich environment of existing and developing class libraries in the areas of GUIs, distributed computing and Grid technologies.

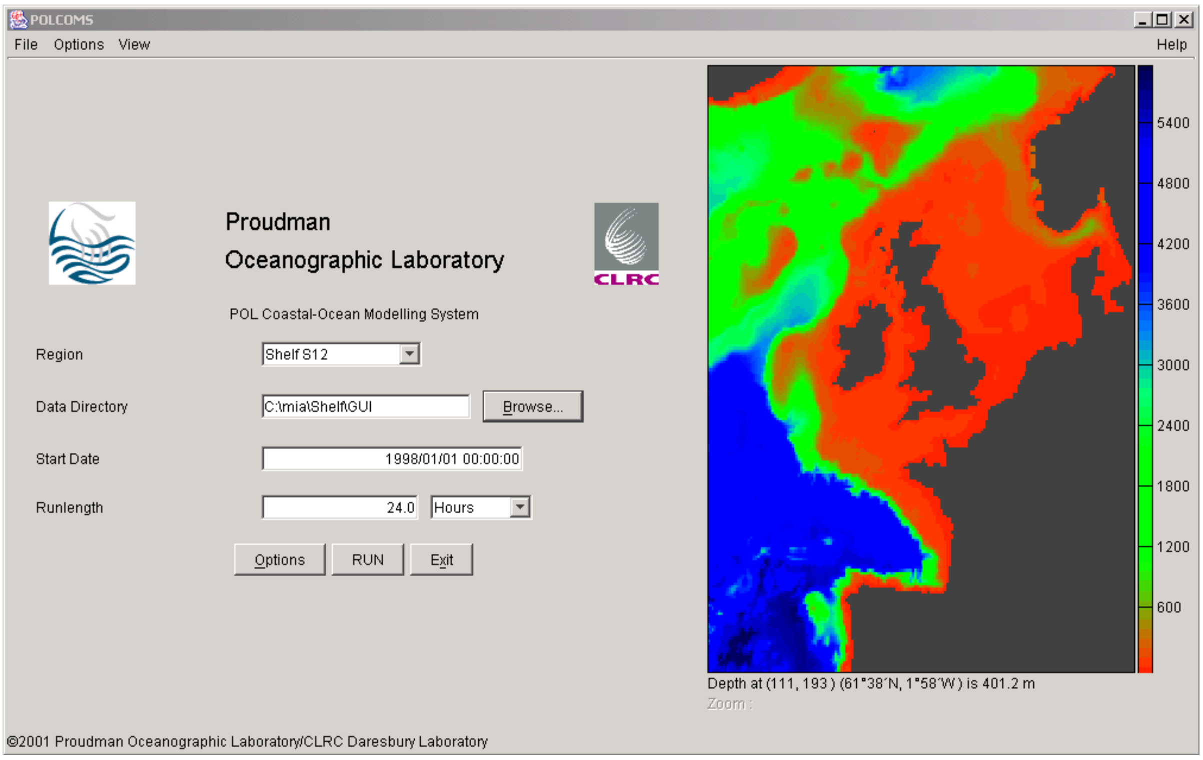
In the emerging Web-based distributed computing environment, we see major benefits in integrating Java with the traditional high-performance programming environment based on Fortran using a client/server model. This vision emphasises the strengths of each component: Java allows easy access to Web-based computing and facilitates the construction of Graphical User Interfaces (GUI); Fortran allows applications best to exploit high-performance hardware.

5.1.3 Design

Although the scope of the current project was limited to producing a GUI for the existing Fortran program running locally, the design gave consideration to likely future developments. We envisage a future scenario in which the model runs as a server on some remote computing resource and the user interacts with a client running on a local PC/workstation. This vision is reflected in a client/server design. The original Fortran program, which reads steering data from files on the local disk, now may be started in a server mode. In server mode it accepts its input parameters from a connection with the client and returns monitor data by the same route. At present this is implemented by wrapping the Fortran code with a number of C routines, which communicate via TCP/IP sockets. A refinement, which may be introduced in the future, is to wrap the model code further with Java and to perform the client/server communication using RMI or CORBA. A further option is to furnish both client and server sides with Globus routines, allowing us to take advantage of Globus' security and remote access features.

5.1.4 Implementation

The present configuration is shown in Figure 4. The Polcoms class contains all the steering and model data for the program and includes methods by which they may be accessed. Other classes extend Polcoms and thereby inherit all the data and methods. Two major classes, PolcomsClient and PolcomsServer, are the basis for the two applications running on the client and server sides, respectively.



Proudman
Oceanographic Laboratory
POL Coastal-Ocean Modelling System

Region Shelf S12

Data Directory C:\mia\Shelf\GUI

Start Date 1998/01/01 00:00:00

Runlength 24.0 Hours

Depth at (111, 193) (61°38'N, 1°58'W) is 401.2 m

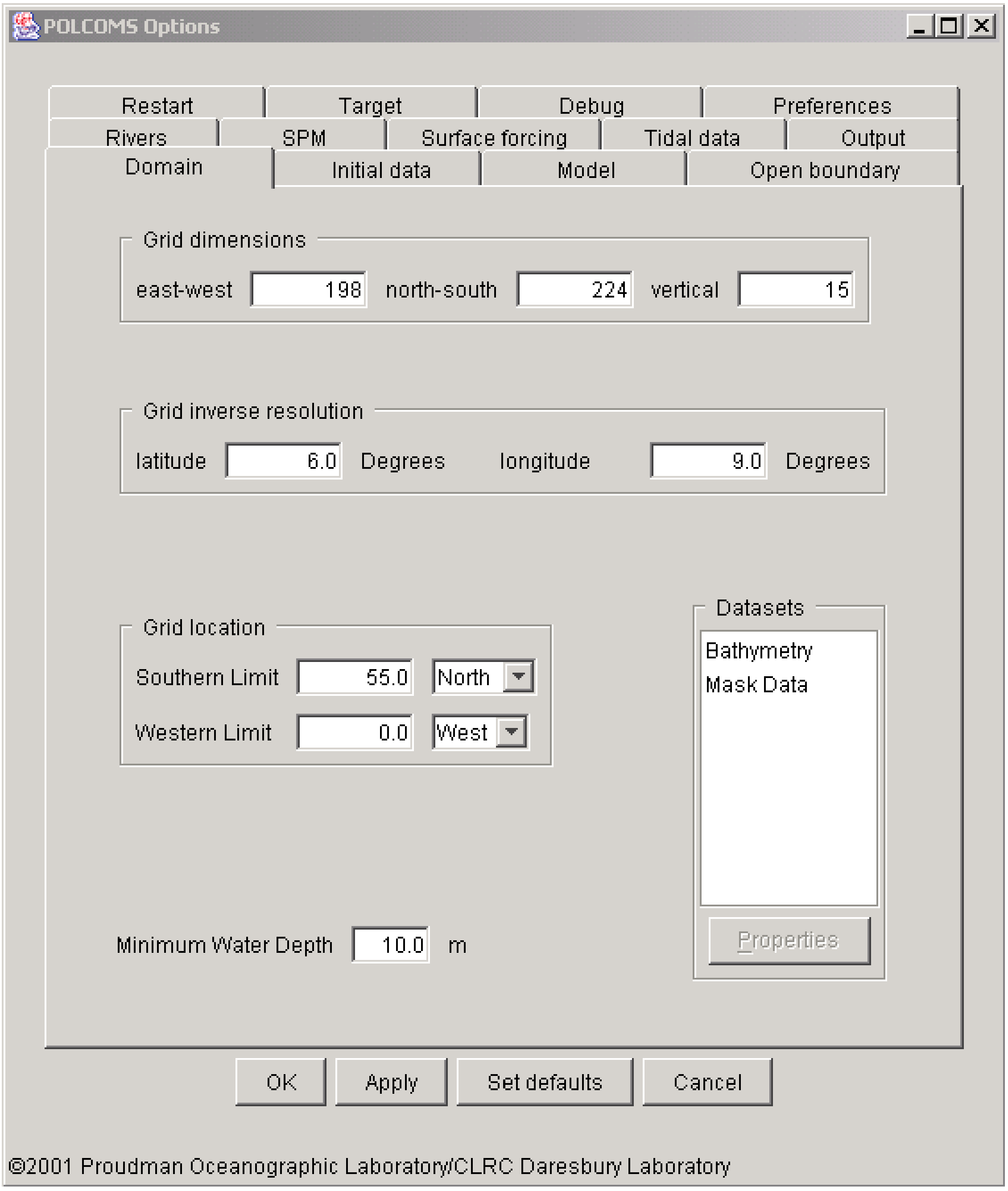
©2001 Proudman Oceanographic Laboratory/CLRC Daresbury Laboratory

Figure 5. The POLCOMS GUI main window, showing the depth map for the currently selected region

The POLCOMS code is naturally highly modular and we can separate its functions into the following: the marine domain, the basic hydrodynamics, tides, open boundaries, initial data, meteorological (surface) forcing, sediments, rivers, output datasets, checkpoint/restart, target system, parallel processing options etc. Each module deserves its own class to hold data, variables and parameters, to contain methods to access and manipulate those data, and to contain methods to display and manage GUI panels to allow those data to be updated.

The main window is shown in Figure 5. A pull down menu allows selection of the region of interest from a library of different regions. Most of the options are available via a set of tabbed panes, with one pane for each of the modules referred to above (Figure 6). When the model is ready to be run, the user presses a big green "Run" button, the GUI communicates the options to the Fortran model code, the model starts and some monitor information is sent back to the GUI for real-time display. The remote connection was to be done using Java RMI, but we were unable to make this work, so we reverted to a quick and easy solution using TCP/IP sockets.

The successes of this project are that we have created a Java GUI linked in real-time to the Fortran application code, after a short learning period we found Java quick, easy and robust to use, and despite being written by an old Fortran *hack*, the code has become more and more object-oriented. Problems include the inability to work with Java RMI. We would like to create a better visualisation interface for input and output data, using one of the freely available visualisation packages, of which VisAD and VTK seem to be the prime candidates. We would also like to use JavaHelp to display HTML help files and documentation.



Grid dimensions

east-west 198 north-south 224 vertical 15

Grid inverse resolution

latitude 6.0 Degrees longitude 9.0 Degrees

Grid location

Southern Limit 55.0 North

Western Limit 0.0 West

Datasets

Bathymetry

Mask Data

Minimum Water Depth 10.0 m

©2001 Proudman Oceanographic Laboratory/CLRC Daresbury Laboratory

Figure 6. The POLCOMS GUI options window, showing a number of tabbed panes with that for the domain options being currently selected.

The DL_POLY Java Graphical User Interface for CCP5

by Bill Smith

DL_POLY is a general-purpose molecular dynamics simulation code that was written to support the research of members of CCP5, the collaborative computational project in molecular simulation. The code has several hundred users in the UK and overseas and a very broad range of applications, e.g. minerals, glasses, proteins. However, owing to its high level of functionality, the code is not easy to use, particularly if one is a novice in simulation methods. A Graphical User Interface has been developed to facilitate access to its capabilities and for checking input data before lengthy simulations commence. The original GUI was written using SDK, the software library from Molecular Simulations Inc, and interfaced with MSI's Cerius 2 visualiser, and for several years this met the needs of many DL_POLY users. However not all users have the MSI software and in some cases wanted to run the code on platforms not supported by MSI. The DL_POLY Java GUI was designed as an alternative.

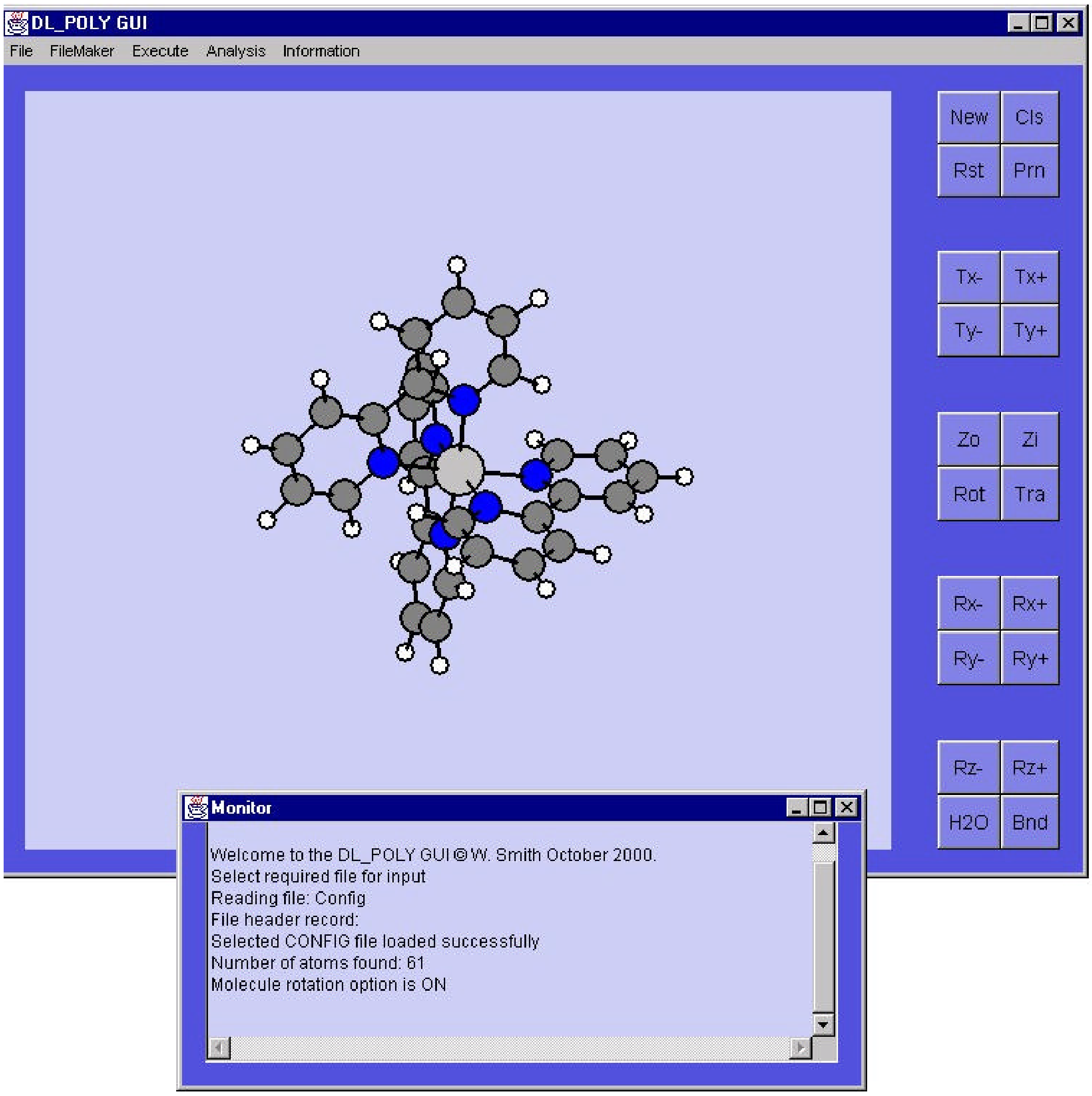
5.1.5 Designing the GUI

Our design for the DL_POLY Java GUI was based on the original SDK version, since this possessed most of the functionality we required. It also meant that we could use many of the FORTRAN routines from our original GUI as the basis of our code. This, of course, meant a translation from Fortran to Java, but this was relatively straightforward, if a little tedious. The main disadvantage however was that this approach meant that we could not exploit the object oriented aspects of Java as fully as we would like, and so by Java standards, our resultant code is inelegant. This is not apparent to the user however, for whom the GUI appears as an efficient and smoothly running package. In time we shall deepen the OO structure of the code to facilitate the import of additional features.

Our first major decision was to implement the GUI as an “application'' and not an “applet". The latter is the more common form of software in Java, as it is used in construction of activated Web pages and the like. However, Java applets require a Web browser to run and are forbidden to make changes to data stored on disc, which was a high priority for our Java GUI. Using Java Server Pages would be an alternative (see below).

Our overall concept of the GUI was that it should consist primarily of a molecular graphics window to render pictures of our molecular systems with various buttons to activate options of different kinds. For example one button could activate the operation to rotate the image, another could send the image to the printer, and so on. We also thought it desirable to have a text display window constantly in view, through which the GUI could relate processing information to the user. This is not strictly necessary, but it had the advantage of keeping separate messages from the GUI and messages from the underlying Java runner, which communicated with the parent X-window.

A third requirement was for some means of selecting more advanced operations, such as generating new molecular configurations and DL_POLY input files, or performing analysis on the results of a simulation. Our idea was that, to a large extent, these should be independent programs (Java classes), with their own control panels and windows, but these should be started up by selecting the appropriate option from a drop-down menu appearing on the top of the Molecular Graphics window. The visual realisation of this concept is shown in Figure 7.



```
Welcome to the DL_POLY GUI © W. Smith October 2000.
Select required file for input
Reading file: Config
File header record:
Selected CONFIG file loaded successfully
Number of atoms found: 61
Molecule rotation option is ON
```

Figure 7. The DLPOLY Java GUI

The menus on the GUI provide most of its functionality. Selecting a menu item with the mouse, in most cases, invokes a separate Java program simply by instantiating the program object. This is a nice and simple technique, requiring a single Java statement to realise. For GUI designers this is liberating, since the desired functionality can be developed and tested independently of the parent GUI and proven to be working correctly before full incorporation. This was how we approached all the support features of the DL_POLY GUI. Each was written independently as a separate application program. The final step we took was to identify common data elements (e.g. arrays and methods) between the different applications and the GUI and to place these elements into a superclass. Each application was then made a subclass of this and the GUI became fully integrated.

Of course in real life things are seldom as simple as this, and it would be deceiving the reader to imply this was accomplished without a small amount of difficulty. However, it is fair to say that most of the difficulty lay in fully understanding the structure of Java, particularly the object orientation concepts of inheritance, event handling and threads, which were alien to Fortran programmers until recently. However, with growing familiarity the capability of building sophisticated interactive programs in a short time becomes a reality.

6 Globus Java CoG Kit

by Daniel Hanlon

Commodity Grid (CoG) toolkits allow the use of Globus Grid services from machines on which Globus might not be installed. They are available in the form of interfaces and libraries that provide GUIs and APIs to access the Globus middleware [10],[12]. The Java CoG kit is the most developed of these projects. It may be downloaded from http://www.globus.org/cog/java/. There are also Python and Perl versions.

Java CoG provides a Globus API in pure Java including the GSI using the IAIK Java SSL libraries to delegate credentials. The CoG kit provides APIs for submitting jobs to Globus gate-keepers, transferring files using GridFTP implemented in Java and querying LDAP servers using the Java Naming and Directory Interface (JNDI).

See http://www.globus.org/security/v1.1/ftp/install.html.

The whole range of Globus services is supported and everything apart from the DUROC support is implemented in Java. This means that with that one exception, all Globus services are accessible from any platform on which there is a Java 2 compliant Virtual Machine i.e. almost all versions of Unix, Windows, MacOS etc.

The intention is that the Java CoG kit implement Globus functionality in a manner so as to integrate well with existing Java standards. Globus job invocation and status callback, for example, is implemented in line with normal Java event handling as shown below.

```
# create a GramJob object gj
GramJob gj = new GramJob(<RSL string>);

# create a callback function which listens to the job
status
gj.addListener(
  new GramJobListener() {
    public void statusChanged(GramJob job) {
      System.out.println("Job: " + job.getRSL() +
"\nStatus: "
        + job.getStatusAsString());

      switch(job.getStatus()) {
        case GramJob.STATUS_FAILED:
        <Do something with> job.getError();
      case GramJob.STATUS_DONE:
        ...;
      case GramJob.STATUS_ACTIVE:
        ...;
      case GramJob.STATUS_PENDING:
        ...;
      case GramJob.STATUS_SUSPENDED:
        ...;
      }
    }
  );

# send the job request
gj.request(<Resource contact string>);
```

Java support has now reached the level where it may be assumed that it is present or at least available for nearly any platform, including Win32. This leads to the possibility that the Java CoG kit may be a suitable tool for adding Grid awareness to many legacy codes. This may be achieved either by using the Java Native Interface (JNI) to link directly with the legacy code, or by the use of files or sockets and launching the virtual machine (JVM) as a separate process. It is being used in the Grid Portal Development Kit (GPDK), see below.

Unfortunately, at the time of writing, there is no CoG kit for the Globus v2.0 release.

6.1 Portal Collaboration and GPDK

by Rob Allan and Daniel Hanlon

The User Portal collaboration was formed as a joint venture for the main US Grid development testbeds at NPACI, NCSA and NASA IPG. The primary goal of the Collaboration is to give computational scientists, researchers and other high performance computer users access to resources via easy to use Web interfaces, see http://www.ipg.nasa.gov/portals/. A set of common components, interfaces and standards is being defined and implemented which will make portal development easier and allow various portals to inter-operate by using the same core infrastructure. These teams, led by Mary Thomas of SDSC, have also formed the GGF Grid Computing Environments (GCE) working group, see http://www.computingportals.org/.

Components developed by the Portal collaboration are being collected in the Java-based Grid Portal Development Toolkit (GPDK) http://dast.nlanr.net/Features/GridPortal/ that uses the Globus Java CoG kit. Practical work on this is principally carried out at NCSA and NASA since SDSC are continuing to develop Perl/CGI based version, GridPort and HotPage, as described above. Portal developers will in the future be able to construct more complex Web portals that can manage user preferences and sessions via Java Server Pages (JSP) http://java.sun.com/products/jsp/ and Java Beans. These provide access to Grid services using Tomcat http://jakarta.apache.org/, an open source Web application server developed by Sun Microsystems as the latest reference implementation of Java Servlets v2.2 and Java Server Pages v1.1, see http://java.sun.com/products/servlet/. GPDK Java beans are grouped into the following five categories: Security; User Profiles; Job Submission; File Transfer; and Information Services.

The GPDK is packaged as a Web application as defined by the Java Servlet 2.2 specification, consisting of HTML Web pages, Java server pages, and Java beans. Tomcat compiles a Java Server Page into a servlet, resulting in HTML that is sent back to the client. Typically, an application instantiates various GPDK Java beans with a desired lifetime using JSP "scriptlets". For instance, user profile beans will persist for the duration of a session (typically 30 minutes, but server configurable), while application specific information may have "application scope" i.e. persistent until the Web server goes down. JSP invokes bean methods to authenticate users, manage profiles, submit jobs, etc.

The GPDK derives most of its functionality from the Globus Java Commodity Grid (CoG) toolkit discussed in Section 6. GPDK Java beans merely present an easier interface for Web developers to use the CoG kit when developing portal server pages.

The use of Java server scripts and beans directly replaces the Perl and C CGI programs and MyProxy delegated certificates which the SDSC group has been using HotPage and GridPort services portals and we have been using in the HPCGrid services portal (see Section 9.2). The overall architecture is however very similar and still rather limited in flexibility. Security implications mean that the expiry time of the delegated proxy credential (or bean) should be very short and a separate password is used. This means that, whilst it would be technically possible to integrate the Web server and browser more closely with Globus, since they both use SSL, it may not be desirable. This is however being tried in the SDSC GridPort client toolkit using cookies. Additional domain-specific plug-in services could be provided using Java, but nothing is available yet.

GPDK is currently being prepared for release under an open source licence and a collaborative agreement between Daresbury and NCSA will enable us to access an early version.

7 DLVizualise

by Barry Searle

DLV v2.1 is based on the AVS Express visualisation system v5.1. CCP3, the collaborative computational project in materials science, has negotiated a deal with AVS allowing us to purchase and distribute licenses for DLV at low cost. CLRC Daresbury Laboratory provides all support for these DLV licenses.

DL Visualize is a graphical user interface for use with a variety of materials simulation software. Based on AVS/Express it is able to display and edit structures periodic in both 2 (surfaces) and 3 (crystals) dimensions. Future versions will also provide interfaces to control, and visualise the output from, surface science applications from the CCP3 program library.

DLV v2.1 includes an interface to CRYSTAL. The final beta version is now available for download from http://www.cse.clrc.ac.uk/Activity/DLV/.

The CRYSTAL program is a joint development by the Theoretical Chemistry Group at the University of Torino and the Computational Materials Science group at CLRC. The program computes the electronic structure of periodic materials within Hartree Fock, density functional or various hybrid approximations [theory]. The Bloch functions of the periodic systems are expanded as linear combinations of atom centred Gaussian functions. Powerful screening techniques are used to exploit real space locality. The code may be used to perform consistent studies of the physical, electronic and magnetic structure of molecules, polymers, surfaces and crystalline solids.

8 GUIs, Scientific Applications and the Grid

Although it is widely stated that the computational Grid and the World Wide Web share the same technology, this is frankly not obvious at first sight.

Web technology is about using http servers and clients to connect to data sources, mostly supplying browsable or downloadable text, the former based on HTML or newer variants with XML headers with server side CGI scripts invoked from forms. The user need not know the geographical location and architecture of the server, although mirror sites exist for the most popular sources. Passwords are not used unless security issues arise, e.g. a customer account such as might exist on a shopping-trolley style portal like http://www.amazon.com/ or http://www.gardening.com/.

Now that Internet technology is the dominant way to obtain information, it is natural to consider how we can use it in a more “interactive" way, as in one definition of e-Science, i.e. to provide “active services". This not only enables the user to load down data or purchase an item with a fixed price but also gives flexible access to data or facilities in a way which must be secure and accountable. Globus [10] contains a set of Perl and shell scripts and a C library and API but does not provide a GUI interface. However, work on the Java Commodity Grid Toolkit (CoG kit), Python CoG kit [15], GridPort Client Toolkit (GCT) and Grid Portal Development Kit (GPDK) are providing some of the required components for future developers. Globus and the Globus Security Infrastructure (GSI) provide a useful Grid infrastructure in the form of a C and shell script toolkit that uses the same underlying software and protocols as the Web. Namely, secure sockets (the Netscape SSL public-private key PKI system proposed as a standard and re-named TLS by IETF) and LDAP (the Netscape Lightweight Directory Access Protocol). Globus can be invoked from the traditional Web Common Gateway Interface (CGI) via HTML forms or directly from a Perl, C or Java interface and forms the basis of control structures required for distributed computing. For further information about the Grid and Web portals, see our user guides [1], [5].

In an effort to construct a common set of components for Grid portal development, the User Portal Collaboration, was formed between the US National Computational Science Alliance Grid (the Alliance or NCSA) and the associated National Laboratory for Network Research (NLANR), based at the University of Illinois at Urbana-Champaign (http://www.ncsa.uiuc.edu/). The San Diego Supercomputer Centre (SDSC) acts as the main hub of the National Partnerships for Advanced Computational Infrastructure (NPACI) Grid http://www.sdsc.edu/ and the NASA Information Power Grid (IPG) http://www.ipg.nasa.gov/. The primary goal of the Grid Portal Collaboration, see http://www.ipg.nasa.gov/portals/, is to allow computational scientists, researchers and other high performance computer users access to resources via an easy to use GUI interface. In Europe the HPC Group of the University of Lecce in Italy is leading portal developments for the SAR Earth Observation community.

Another project is the Cactus Grid environment for computational cosmology [7] and the HPCGrid portal developed at Daresbury Laboratory [1].

Exemplars of services expected from a GUI to control distributed computing applications include:

- single point of login to a geographically distributed environment, with session control providing a pervasive and persistent workspace;
- transferring files, including file upload, file download, and third party file transfers (migrating files between various storage systems);
- running simulations either interactively or submitted to a batch queue on a remote system;
- querying databases for resource/ job specific information;
- running visual data analysis or using computational steering systems.

9 HTML, Web forms, C and Perl/CGI

by Rob Allan

A GUI built from HTML requires a Web browser and also a server to interpret the commands which basically request download of information pages via HTTP GET and run scripts based on input forms using HTTP POST and the environment-based Common Gateway Interface (CGI). If only HTTP GET is required the whole thing can be supplied on CD, as was done with the Grid Support Centre Grid Starter Kit (GSC/GSK) http://esc.dl.ac.uk/StarterKit/ and annual Daresbury Machine Evaluation Workshops http://www.cse.dl.ac.uk/Activity/DisCo/. `file:/` can then be used instead and the CD can be made self-starting on Windows systems and Linux with the KDE desktop environment and Konqueror Web browser.

A self-starting win32 CD must be packaged using `mkisofs` to create an ISO image with the options:

```
mkisofs ...
```

It should include the files `index.htm`, which is a copy of UNIX file `index.html`, `autorun.inf` and executable of `start.exe` which can be copied from an existing win32 system. The file `autorun.inf` looks like this:

```
autorun
start = start.exe index.htm
```

We have used the following packages for server-based Web portal development based on Globus and CGI. They are all distributed as open source:

- Apache v1.3.14 from http://httpd.apache.org/ and Apache secure server from http://httpd.apache-ssl.org/
- OpenSSL v0.9.6 (previously SSLeay) by Eric A. Young from http://www.openssl.org/. See also http://www.ultranet.com/~fhirsch/Papers/wwwj/article.html/
- OpenLDAP v1.2.7 OpenLDAP with Globus patches from http://www.OpenLDAP.org/
- Perl v5.6 from http://www.perl.com/CPAN/
- Perl CGI.pm v2.74 from http://www.perl.com/CPAN/
- CGIC from http://www.boutell.com/
- SDCS GridPort toolkit from http://gridport.npaci.edu/
- SDCS HotPage toolkit from http://hotpage.npaci.edu/
- User Portal Collaboration MyProxy toolkit from http://dast.nlanr.net/Features/MyProxy/
- Perl Extend.pm, Stty.pm, Tty.pm from http://www.perl.com/CPAN/

9.1.1 The Apache HTTP server

Apache is a popular Web server that can easily be installed on a variety of UNIX and other hosts. It can be downloaded from http://www.apache.com/. Binary distributions are provided, but these can be very system dependent as they try to use “shared object" libraries which are dynamically loaded. However the source is distributed too and can be compiled using the, now *de facto*, GNU configure procedure. Let's assume that we have down-loaded and expanded Apache v1.3.14 into a directory `/usr/local/Apache/apache-1.3.14`:

```
cd /usr/local/Apache/apache-1.3.14 ./configure
--prefix-/usr/local/apache make make install cd
/usr/local/apache/bin
apachectl start
```

The last commend should start the `httpd` daemon on this host. It uses the file `/usr/local/apache/bin/httpd.conf`. This file may have to be edited if the shared objects are not being used to remove references to dynamically loaded modules. See also above.

A number of directories are created, the most important of which are:

```
/usr/local/apache/htdocs
/usr/local/apache/bin
/usr/local/apache/conf
/usr/local/apache/cgi-bin
```

The first is the root of the Web-accessible file system visible on the server; the last is the directory where scripts and programs that are to be invoked via the CGI interface must be installed by default. These must have suitable permissions for execution as “nobody" (including the directory). The server program is in `bin` and is started as follows:

```
cd /usr/local/apache/bin; apachectl start
```

It will probably be necessary to change the default configuration to enable CGI scripts to be installed in other places and to enable server-parsed `http://` (shtml). This is done by un-commenting the following lines in `conf/httpd.conf`:

```
AddHandler cgi-script .cgi AddType text/html .shtml
AddHandler
server-parsed .shtml
```

To password protect certain directories underneath `htdocs` it is possible to add to `access.conf` as follows:

```
<Location> Options Includes ExecCGI </Location>

<Location /HPCPortal/secure> AuthName grid AuthType Basic
AuthUserFile
    /usr/local/apache/etc/passwd AuthGroupFile
    /usr/local/apache/etc/group <Limit GET POST> require
valid-user
    require group grid </Limit> </Location>
```

This will require a basic UNIX id and password to be entered into the browser via a default window if the secure directory is referenced.

After editing any configuration file the server must be restarted for example as follows:

```
tci18.dl.ac.uk>ps -Uroot | grep httpd 0 43690 - 4:45 httpd
kill -HUP
   43690
```

9.1.2 The Perl/CGI and C/CGIC Interfaces

The Common Gateway Interface (CGI) is the standard way to get executable programs and scripts to run as UNIX id “nobody" on a Web server by submitting HTML forms completed on a Web browser. Parameters are passed from the browser as environment variables when an HTTP POST action is executed. Globus is the toolkit which allows you to run programs as “somebody" on a machine other than the Web server. The MyProxy toolkit [18] provides the necessary security interface between CGI and Globus.

CGI provides a means to access data typed in HTML form fields and send a reply via stdout to the HTTP server with an XML header. A variety of routines to handle MIME types are provided for the reply. HTML input can be in a variety of forms that include single-choice menus, multi-choice, single-input form boxes, expandable text boxes, radio buttons and multiple-choice buttons. Name-value pairs defined in this input become part of the environment of the CGI script.

In many Web applications the Perl v5.6 scripting language is used for CGI programming because of the ease with which text strings can be manipulated and system commands called. CGIC v1.07 is an ANSI C library for CGI programming. The source was written by Thomas Boutell and is downloadable from http://www.boutell.com/. There is comprehensive and easily readable documentation with some programming examples which process a variety of example Web forms. Perl has the advantage that development is more rapid since it can be interpreted whereas C must be compiled.

A CGIC server program must be called `cgiMain()` as the entry `main()` is provided by the C library. As well as a set of routines to extract the text fields sent from the HTML form, the usual environment variables are available as C variables. This is also true in Perl.

In our HPCGrid Services Portal, C is used for operations involving Globus and Perl is used otherwise.

9.2 HPCGrid Services Portal

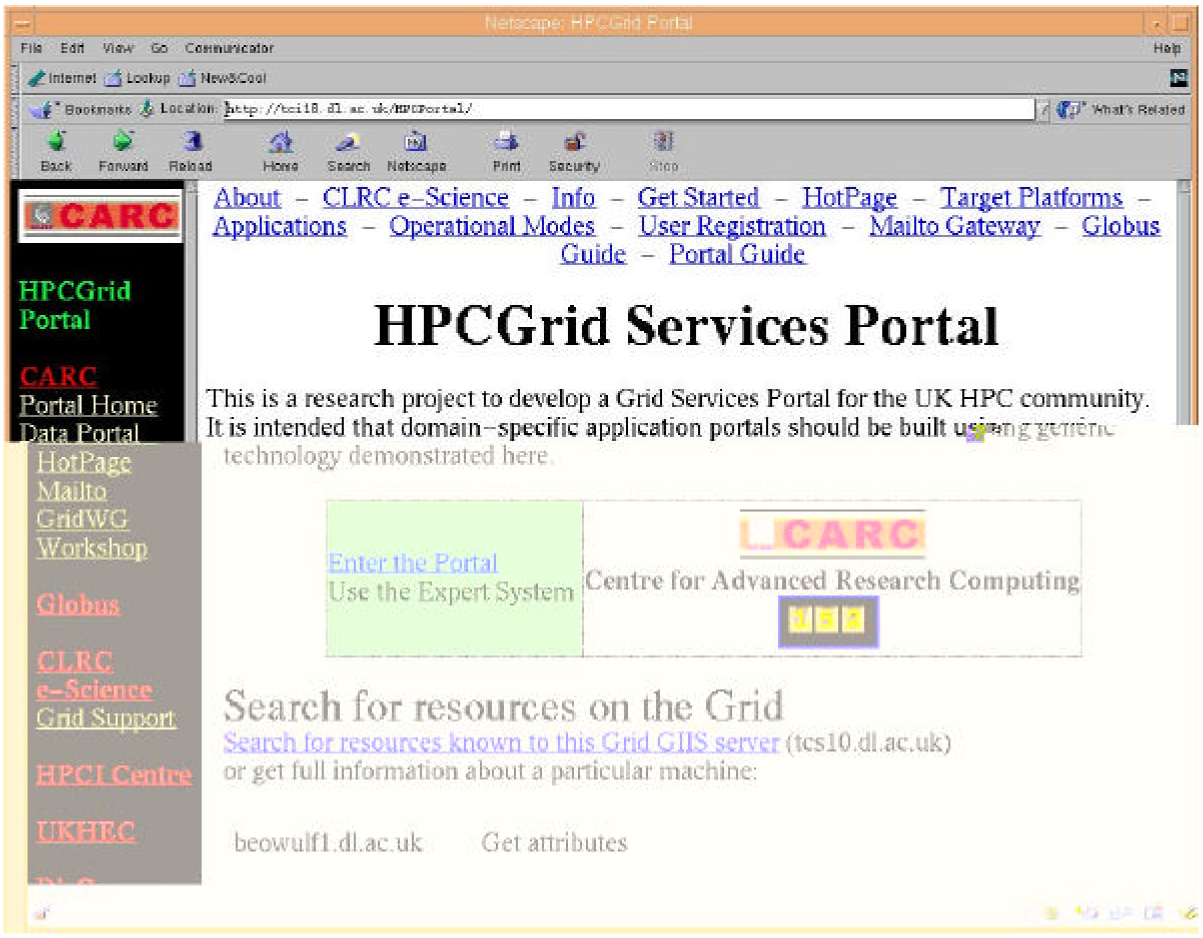
The aim of the HPCGrid Portal is to provide users of Grid-enabled HPC resources with a suite of Web- based tools to discover and select target systems, transfer files and run jobs on remote machines. It also provides a single login facility from a Web browser. This is enabled by using the MyProxy software developed as part of the Grid Portals Collaboration between NCSA at the University of Illinois at Urbana-Champaign and the San Diego Supercomputer Centre, SDSC. MyProxy is a very secure way of accessing Grid certificates and it requires a secure server to be set up. Proxy certificates are issued which can then be delegated to a Web server using a new password. Because they have a limited lifetime, they will be of little use to anyone except the owner if the Web server is hacked. We believe security of this standard this is very important if the Grid is to be used in a routine way. It is likely that, possibly overlapping, thematic or departmental "mini-Grids" will emerge first, and will combine as groups ("virtual organisations" in the Grid language) begin to trust one another.

Once logged onto the Grid the user is presented with a "desktop" capability and the following functionality:

- resource discovery with specified attributes, e.g. machine type, memory size, etc.;
- creation of a temporary file space on a remote machine;
- file transfer;
- running remote commands and submitting jobs;
- automatic retrieval of "stdout" and "stderr" files onto the desktop;

- automatically maintains a list of selected machines and remote environment variables on the desktop for subsequent use from the same browser IP address.

Currently this functionality is invoked using "old fashioned" HTML and CGI with C and Perl server-side programs and include files. It would however be relatively straightforward to re-produce this functionality using Java if this proves to be the best way forward. Evaluation of the Java Commodity Globus Toolkit (CoG Kit) is currently under way, and we note that Edinburgh are using Java for the MDH e-Portal and Daresbury are using Java for DLVizualise which is a portal for a number of applications and databases for materials science research.


Netscape: HPCGrid Portal
File Edit View Go Communicator Help
Internet Lookup New&Cool
Bookmarks Location: http://tci18.dl.ac.uk/HPCPortal/
What's Related
Back Forward Reload Home Search Netscape Print Security Stop

About – CLRC e-Science – Info – Get Started – HotPage – Target Platforms – Applications – Operational Modes – User Registration – Mailto Gateway – Globus Guide – Portal Guide

CARC

HPCGrid Portal

CARC
Portal Home
Data Portal
HotPage
Mailto
GridWG
Workshop

Globus

CLRC
e-Science
Grid Support

HPCI Centre

UKHEC

HPCGrid Services Portal

This is a research project to develop a Grid Services Portal for the UK HPC community. It is intended that domain-specific application portals should be built using generic technology demonstrated here.

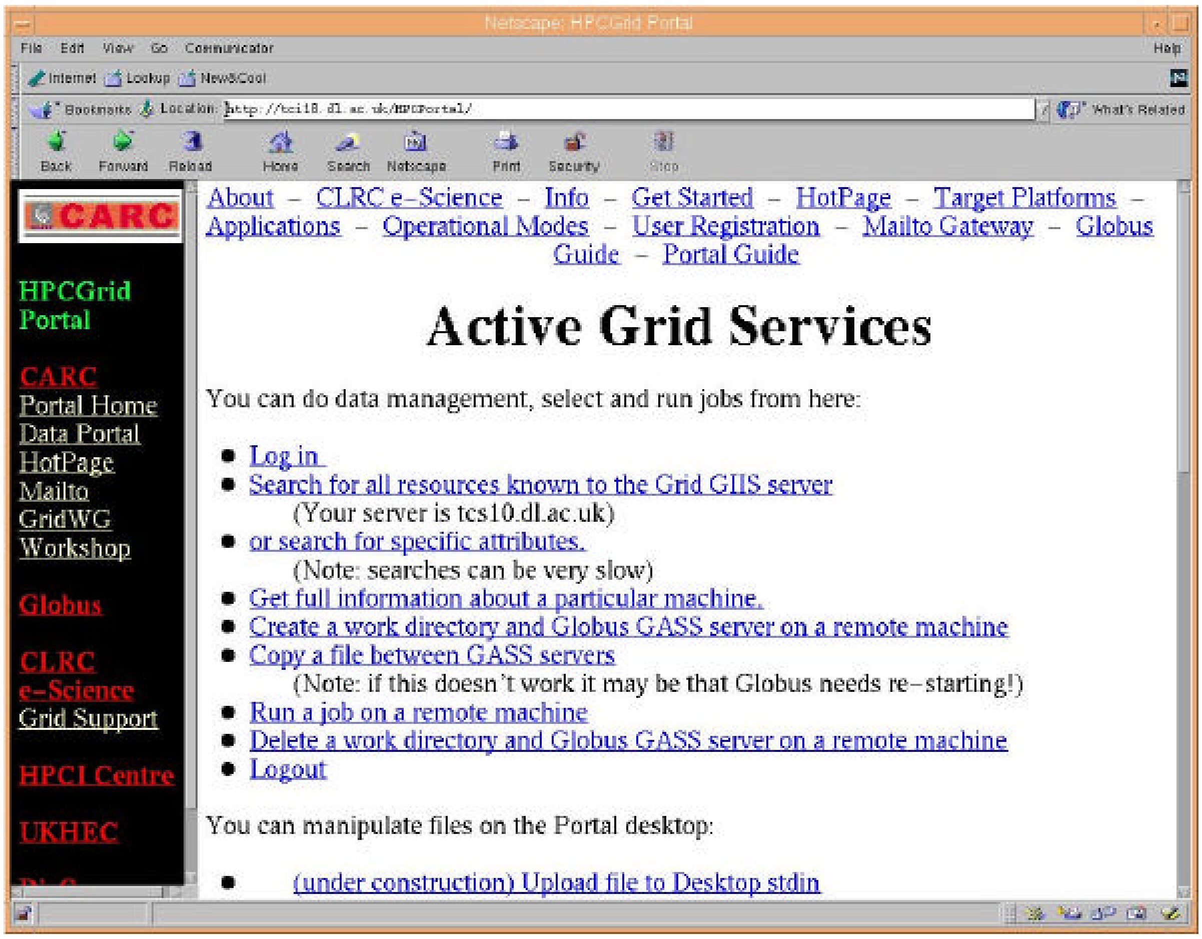
Enter the Portal Use the Expert System	CARC Centre for Advanced Research Computing

Search for resources on the Grid

Search for resources known to this Grid GIIS server (tcs10.dl.ac.uk)
or get full information about a particular machine:

beowulf1.dl.ac.uk Get attributes

Figure 8. The HPCGrid Portal



About – CLRC e-Science – Info – Get Started – HotPage – Target Platforms – Applications – Operational Modes – User Registration – Mailto Gateway – Globus Guide – Portal Guide

Active Grid Services

You can do data management, select and run jobs from here:

- Log in
- Search for all resources known to the Grid GIIS server
  (Your server is tcs10.dl.ac.uk)
- or search for specific attributes.
  (Note: searches can be very slow)
- Get full information about a particular machine.
- Create a work directory and Globus GASS server on a remote machine
- Copy a file between GASS servers
  (Note: if this doesn't work it may be that Globus needs re-starting!)
- Run a job on a remote machine
- Delete a work directory and Globus GASS server on a remote machine
- Logout

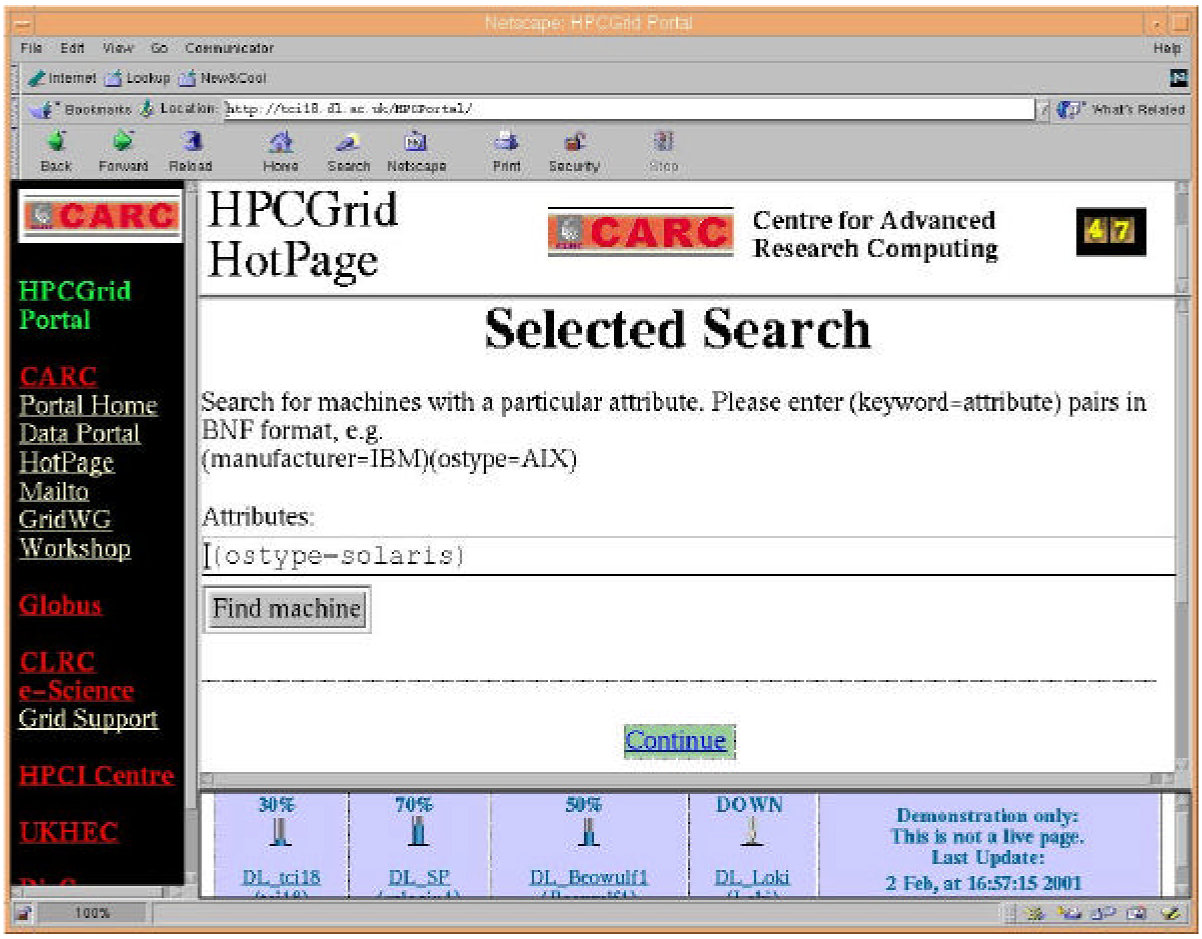
You can manipulate files on the Portal desktop:

- (under construction) Upload file to Desktop stdin

Figure 9. The HPCGrid Services

Other work going on in the CLRC e-Science Centre may be of interest to HEC Users. In particular the Data Portal will be integrated with the HPCGrid Portal. This will enable linking of computational resources with a range of experimental and modelling databases for research in chemistry and physics. It is currently linked to prototype facilities on the SRS at Daresbury and ISIS at Rutherford Appleton Laboratory. We believe experimentalists may also wish to carry out simulations on HEC resources to support their scientific research and a Portal with an in-built expert system is a target for future work in this area. Other plans include re-design of the user profile management in the portal server.

For better resource monitoring and utilisation we have added the HotPage software from the San Diego Supercomputer Centre (SDSC) to the HPCGrid Portal. With the use of JavaScript and remote cron scripts and http to collect resource information, HotPage provides a comprehensive view of the Grid. Icons that indicate availability and load of remote systems are dynamically updated. On-line registration and user manuals are provided for each system, with the system manager's co-operation. HotPage is currently being re-written to fully utilise the capabilities of Globus and the underlying LDAP information register.


Netscape: HPCGrid Portal
File Edit View Go Communicator Help
Internet Lookup New&Cool
Bookmarks Location: http://tci18.dl.ac.uk/HPCPortal/ What's Related
Back Forward Reload Home Search Netscape Print Security Stop
CARC
HPCGrid Portal
CARC
Portal Home
Data Portal
HotPage
Mailto
GridWG
Workshop
Globus
CLRC
e-Science
Grid Support
HPCI Centre
UKHEC
CARC Centre for Advanced Research Computing
100%

HPCGrid HotPage

Selected Search

Search for machines with a particular attribute. Please enter (keyword=attribute) pairs in BNF format, e.g.
(manufacturer=IBM)(ostype=AIX)

Attributes:
(ostype=solaris)

Find machine

Continue

30%	70%	50%	DOWN	Demonstration only: This is not a live page. Last Update: 2 Feb, at 16:57:15 2001
DL_tci18	DL_SP	DL_Beowulf1	DL_Loki	

Figure 10. Hotpage with HPCGrid Portal

Our Portal can therefore also provide access to a wealth of on-line information and a user registration service which is planned as a future project so that and a potential user may submit a request for an id on any connected platform and automatically receive a Grid Certificate too. Of course the local system managers have ultimate control and a peer review mechanism is included as on the current CSAR service.

Even without a Grid id it is possible to examine what resources are available by accessing the search facility. The more selective search and job submission is however only possible once you are logged in.

Whilst the prototype is currently only visible from within the CLRC firewall, but a new server is being commissioned and it will be visible to everyone soon. All the documents we have developed are available via http://esc.dl.ac.uk/GridWG/.

10 Acknowledgements

The preparation of this report, and the work described, was partly funded by EPSRC through grant GR/N09688 to the UKHEC Collaboration at CCLRC Daresbury Laboratory and the Universities of Edinburgh and Manchester. They were also partly funded by the CCLRC e-Science Centre and the Service Level Agreement between EPSRC and the CSE Department which funds support for the CCPs.

11 References

[1] R.J. Allan, *Developing a Web Portal for the Computational Grid*, Draft Technical Report, UKHEC Collaboration and CLRC e-Science Centre (2001), http://esc.dl.ac.uk/GridWG

[2] R.J. Allan and M. Ashworth, *A Survey of Distributed Computing, Grid Computing, Meta-computing and Network Information Tools*, UKHEC Report, 2001, http://www.ukhec.ac.uk/reports/survey.pdf

[3] R.J. Allan, D.R.S. Boyd, T. Folkes, C. Greenough, D. Hanlon, R.P. Middleton and R.A. Sansum, *Evaluation of Globus and Associated Grid Middle-ware*, Report to the DTI (CLRC e-Science Centre, May 2001), http://esc.dl.ac.uk/StarterKit/

[4] R.J. Allan, D.R.S. Boyd, T. Folkes, C. Greenough, D. Hanlon, R.P. Middleton, R.A. Sansum, Elson Mourão, Rob Baxter, David Henty, Mark Parsons, John Brooke, W.T. Hewitt, Mike Daw, Jon MacLaren, Jon Gibson, Graham Riley and Stephen Pickles, *Globus and Associated Grid Middleware. Consolidated Evaluation Report from UKHEC Sites*, (UKHEC, December 2001), http://esc.dl.ac.uk/StarterKit/

[5] R.J. Allan, D. Hanlon, R.F. Fowler and C. Greenough, *A Globus Developers' Guide with Installation and Maintenance Hints*, Draft Technical Report, UKHEC Collaboration and CLRC e-Science Centre (2001) http://www.dl.ac.uk/TCSC/UKHEC/GridWG/

[6] D.M. Beazley, *Python Essential Reference*, (New Riders, Indianapolis, 2000) ISBN 0-7357-0901-7

[7] The Cactus Computational Toolkit, http://www.cactuscode.org/

[8] M. Conway, *A Tkinter Life Preserver*, University of Virginia, http://www.python.org/doc/life-preserver/index.html

[9] Warren DeLano Inc, *PyMOL*, http://pymol.sourceforge.net/

[10] I. Foster and C. Kesselman, *GLOBUS: a Metacomputing Infrastructure Toolkit,* Int. J. Supercomputing Applications (1997) 115-28

[11] I. Foster and C. Kesselman (eds.), *The Grid: Blueprint for a new Computing Infrastructure,* (Morgan Kaufmann Publishers, 1998) ISBN 1-55860-475-8. Abstracts of chapters and ordering information from http://www.mkp.com/Grids/.

[12] I. Foster and K. Kesselman, *The Globus Project: a status report*, IPPS/SPDP'98 Heterogeneous Computing Workshop S.4-18 (1998), http://www-fp.globus.org/documentation/papers.html

[13] I. Foster, C. Kesselman and S. Tuecke, *The Anatomy of the Grid: Enabling Scalable Virtual Organisations*, Int. J. Supercomputing Apps. (2001), http://www.globus.org/research/papers/anatomy.pdf

[14] J.E. Grayson, *Python and Tkinter Programming*, (Manning, Greenwich, Connecticut, 2000) ISBN 1884777813

[15] K.R. Jackson, *pyGlobus*, Lawrence Berkeley National Laboratory, http://www-itg.lbl.gov/grid/projects/pyGlobus.html

[16] S. Lumholt and G. van Rossum, *Tkinter*, http://www.python.org/topics/tkinter/

[17] G. McFarlane and P. Munnings *Pmw*http://pmw.sourceforge.net/

[18] MyProxy Online Credential Repository, http://grid.ncsa.uiuc.edu/myproxy/

[19] J.K. Ousterhout, *Tk*, http://www.scriptics.org/

[20] J.K. Ousterhout, *Tcl and the Tk Toolkit*, (Addison-Wesley, Reading, Massachusetts, 1994) ISBN 0-201-63337-X

[21] G. van Rossum, *Python*, http://www.python.org/

[22] L.D. Stein, *How to set up and maintain a Web Server*, (Addison-Wesley, 1997-2000) ISBN 0-201-63462-7

Appendix A - Layout Management in tk

One of the easiest ways to start GUI programming is to use Scriptics' Tk and Tkinter wrappers. There are a lot of similarities to the curses library. Tk and curses have a surprisingly similar Python interface, despite the fact that curses is intended to develop text consoles and Tk to implement GUIs. Before using either library, you need a basic understanding of windows and event loops and a reference to the available widgets.

Tkinter comes with many Python distributions, so support libraries or other Python modules need not be downloaded. Collections of higher-level user interface widgets are mentioned below, but you can do a lot with Tkinter itself, including construction of your own high-level widgets. Learning the base Tkinter module will introduce you to the Tk way of thinking, which is important even if you go on to use more advanced widget collections.

Both TK and the Tkinter wrappers are well designed, user-friendly, and one of the easiest ways to get started with GUI programming.

There are a number of things to note in widget programming:

- Every widget has a parent. Whenever a widget is created, the first argument to the instance creation is the parent of the new widget;
- If there are any other widget creation arguments, they are passed by name. This Python feature gives a lot of flexibility in specifying options or allowing them to default;
- A number of widget instances are global variables. We could make them local by passing them from function to function in order to maintain a theoretical purity of scope, but it would be much more trouble than it is worth. Besides, making these basic interface elements *global* underlines the fact that they are useful in all of our functions. But be sure to use a good naming convention for your own global variables;

After we create a widget, we call a *geometry manager* method to let Tk know where to put it. A lot of magic goes into Tk's calculation of the details, especially when windows are resized or when widgets are added dynamically. In all cases we need to let Tk know which set method to use.

Tk provides three geometry managers: `pack(), grid() and place()`. The former are the most commonly used and `place()` can be used for more fine-grained control. Most of the time you'll use `pack()`.

It is possible to call the `pack()` method without arguments. If this is done there is no control over layout on the canvas so some hints should be given. The most important of these will then be the `side` argument. Possible values are LEFT, RIGHT, TOP, and BOTTOM, which are variables in the Tkinter namespace.

More sophisticated uses of `pack()` make use of the fact that widgets can be nested. In particular, the `Frame` widget does little more than act as a container for other widgets, although it can also show borders of various types. It is useful to use several frames in the desired orientations and then add other widgets within each frame. Frames (and other widgets) are packed in the order their `pack()` methods are called. So if two widgets both ask for `side=TOP`, it's first come, first served.

The `grid` geometry manager overlays a parent widget with invisible graph-paper lines. When a widget calls `grid(row=3,column=4)`, it's requesting of its parent that it be placed on the third row and on the fourth column. The number of rows and columns of the parent are computed by looking at the requests made by all its children.

It is important to use a geometry manager for all widgets, otherwise they may not appear on the display.

Tkinter also makes menus painless. In addition to simple text, it is possible to populate menus with different fonts, pictures, checkboxes, and all sorts of fancy child widgets. A simple example of a menu is:

```
menu_frame.tk_menuBar(file_menu(),            action_menu(),
help_menu())
```

Most of the work that must be done lives in the functions called `*_menu()`. A typical drop-down menu might be created as follows:

```
def helpMenu():
    myHelp = Tkinter.Menubutton(menu_frame, text='Help',
underline=0)
    myHelp.pack(side=Tkinter.LEFT, padx="2m")
    myHelp.menu = Tkinter.Menu(myHelp)
    myHelp.menu.add_command(label="How To", underline=0,
command=HowTo)
    myHelp.menu.add_command(label="About", underline=0,
command=About)
    myHelp['menu'] = myHelp.menu
    return myHelp
```

A drop-down menu is thus a Menubutton widget with a Menu widget as a child. The Menubutton is `pack()`'d to the appropriate location (or `grid()`'d, etc.), and the Menu widget has items added with the `.add_command()` method as above.

Obtaining user input is also straightforward. The example below shows how the Label widget displays output. The basic widget for field input is Entry. It's simple to use, but the technique might be a bit different from what you might expect if you've used Python's `raw_input()` or curses' `getstr()` methods before. Tk's Entry widget does not return a value that can be assigned. It instead populates the field object by taking an argument. The following function, for example, allows the user to specify an input file.

```
def getSource():
   myWindow = Tkinter.Toplevel(root)
   myWindow.title('Source File?')

Tkinter.Entry(myWindow,width=30,textvariable=source).pack()
   Tkinter.Button(myWindow, text="Change",
                  command=lambda: update_specs()).pack()
```

There are a few things to notice at this point. We've created a new Toplevel widget and a dialog box for this input, and we've specified the input field by creating an Entry widget with a `textvariable` argument. In addition the `textvariable` argument does not specify a simple string variable. Instead it refers to a StringVar object. In our case, the `init_vars()` function that was called from `main()` contains these lines.

```
source = Tkinter.StringVar()
source.set(‘myFile.txt’)
```

This creates an object suitable for taking user input and gives it an initial value. This object is modified immediately every time a change is made within an Entry widget that links to it. The change occurs for every keystroke within the Entry widget in the style of `raw_input()`, and not just upon termination of a read.

To do something with the value entered by the user, the `get()` method can be used, for example:

```
source_string = source.get()
```

The techniques outlined here, along with the ones used in the full application source code described below, should get you started with Python/Tk and Tkinter programming. After you play with it a bit you'll find that it's not hard to work with. One advantage is that many languages other than Python may access the Tk library, so what you learn using Python's Tkinter module is mostly transferable to other languages.

Appendix B - Further reading, on-line tutorials and references

For a good starting point see http://python.org/topics/tkinter/ the python organisation's Tkinter Resource.

Several widget collections may save you time in constructing complex GUIs http://python.org/topics/tkinter/widgets.html. PMW (Python Mega Widgets) [17] is written in 100\% Python and is widely used in the Python community. See our example below.

Fredrik Lundh has written a tutorial for Tkinter that contains much more detail than covered here.

See http://www.pythonware.com/library/tkinter/introduction/index.htm

A few printed books are worth checking out. The first is an introduction to TK itself, the second is specific to Python and has many examples that use the PMW collection:

TCK/TK in a Nutshell, by Paul Raines and Jeff Tranter (O'Reilly, 1999)

Python and Tkinter Programming by John E. Grayson (Manning, 2000)

http://activestate.com/Products/ActivePython/Download.html ActiveState has made available a Python distribution that includes Tkinter and a variety of other useful packages and modules not contained in most other distributions (they also have an ActivePerl distribution for those inclined towards that other scripting language).

http://www.scriptics.com/ Scriptics (home of the maintainers and creators of TK) has been renamed.

Read these related articles by David Mertz on *developerWorks*:

http://www-106.ibm.com/developerworks/linux/library/lpython6.html?dwzone=linux *Charming Python: Curses programming in Python*

http://www-106.ibm.com/developerworks/library/python3.html *Charming Python: My first Web-based filtering proxy*

Take a look at the http://gnosis.cx/download/charming_python_9.zip

http://www.hetland.org/python/instant-hacking.php *Instant Hacking: Learn how to program with Python* by Magnus Lie Hetland.

http://www.honors.montana.edu/~jjc/easytut/easytut/ *A Non-Programmer's Tutorial for Python* by Josh Cogliati.

http://www.freenetpages.co.uk/hp/alan.gauld/ *Learning to Program*. An introduction to programming for those who have never programmed before, by Alan Gauld. It introduces several programming languages but has a strong emphasis on Python.

http://www.ibiblio.org/obp/thinkCSpy/ *How to Think Like a Computer Scientist* The Python version of Allen Downey's http://www.ibiblio.org/obp/thinkCS.html open source book with Jeff Elkner. Hosted by the http://www.ibiblio.org/obp/ Open Book Project at http://www.ibiblio.org/ Ibiblio — the public's library.

(Also at http://www.andamooka.org/reader.pl?section=thinkpython Andamooka.)

http://www.livewires.org.uk/python/ *LiveWires*. A set of Python lessons used during 1999 and 2000 children's summer camps in Britain by Richard Crook, Gareth McCaughan, Mark White, and Rhodri James. Aimed at children 12-15 years old.

http://www.phrantic.com/scoop/tocpyth.htm *Learn to Program using Python*. A tutorial by Richard Baldwin.

http://www.amazon.com/exec/obidos/ASIN/0201709384/pythonbookstore *Learn to Program using Python* by Alan Gauld, author of one of the http://www.freenetpages.co.uk/hp/alan.gauld/ above tutorials.

http://www.amazon.com/exec/obidos/ASIN/0672317354/pythonbookstore *Teach Yourself Python in 24 Hours* by Ivan van Laningham.

The PyMOL environment with all the libraries involved forms a quite extensive system. Here is the material used in this project:

- *Python Essential Reference* by D.M. Beazley [6]

  A pure Python reference book, it has been build up in such a way that the level of detail increases from the front to the back. The advantage is that the beginner does not drown in intricate details, the drawback is that one subject is discussed in more than one place.

- *Tcl and the Tk toolkit* by J.K. Ousterhout [20]

  This book contains a very detailed description of the Tk toolkit that underlies Tkinter. As such it is useful to learn about Tk's capabilities, however it doesn't discuss the particulars of the Python binding used to build Tkinter.

- *A Tkinter Life Preserver* by M. Conway [7]

  A paper outlining everything about how Tkinter relates to Tk, and how to use Tkinter from Python. This paper was purely meant as a stop gap effort in the time when no Tkinter specific documentation existed. However, if you are predominantly using Pmw then this life preserver is just enough to be able to fill the functionality gaps with Tkinter.

- *Python and Tkinter Programming* by J.E. Grayson [12]

  This book specifically addresses the subject of writing GUIs in Python using Tkinter and Pmw. It supersedes the previous two sources of information.

- *Python megawidgets* by G. McFarlane [17]

  This manual comes with the Pmw distribution. Together with the examples that come with the distribution as well it provides most information needed to develop a simple GUI. In particular if the Pmw library covers the functionality you need then this documentation may be all you need.