Astronomical Data Analysis Software and Systems VII
ASP Conference Series, Vol. 145, 1998
R. Albrecht, R. N. Hook and H. A. Bushouse, eds.

Grid OCL : A Graphical Object Connecting Language

I. J. Taylor[1]

Department of Physics and Astronomy, University of Wales, College of Cardiff, PO BOX 913, Cardiff, Wales, UK, Email: Ian.Taylor@astro.cf.ac.uk

B. F. Schutz[2]

Albert Einstein Institute, Max Planck Institute for Gravitational Physics, Schlaatzweg 1, Potsdam, Germany. Email: schutz@aei-potsdam.mpg.de

Abstract. In this paper, we present an overview of the Grid OCL graphical object connecting language. Grid OCL is an extension of Grid, introduced last year, that allows users to interactively build complex data processing systems by selecting a set of desired tools and connecting them together graphically. Algorithms written in this way can now also be run outside the graphical environment.

1. Introduction

Signal-processing systems are becoming an essential tool within the scientific community. This is primarily due to the need for constructing large complex algorithms which would take many hours of work to code using conventional programming languages. Grid OCL (Object Connecting Language) is a graphical interactive multi-threaded environment allowing users to construct complex algorithms by creating an object-oriented block diagram of the analysis required.

2. An Overview

When Grid OCL is run three windows are displayed. A *ToolBox* window, a *MainGrid* window and a *Dustbin window* (to discard unwanted units). Figure 1 shows the *ToolBox* window which is divided into two sections. The top section shows the available toolboxes (found by scanning the toolbox paths specified in the Setup menu) and the bottom shows the selected toolbox's contents. Toolboxes (and associated tools) can be stored on a local server or distributed throughout several network servers. Simply adding the local or *http* address in the toolbox and tool path setup allows on-the-fly access to other people's tools.

[1]A post doctoral programmer at Cardiff who has been developing Grid OCL since January 1996.

[2]Professor Schutz is a Director of the Albert Einstein Institute, Potsdam

© Copyright 1998 Astronomical Society of the Pacific. All rights reserved.

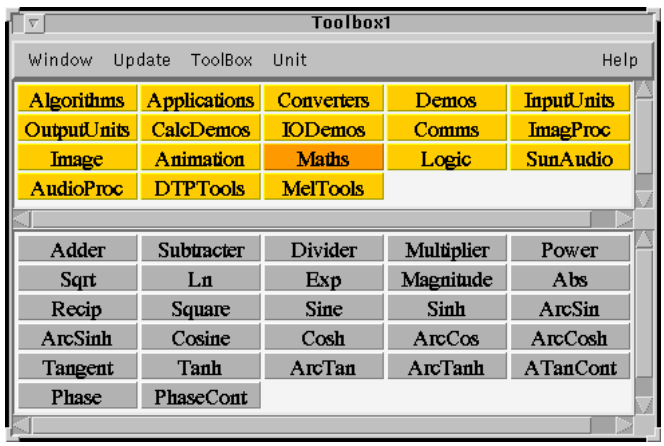

Figure 1. Grid OCL's ToolBox window. Toolboxes can be organised in a similar way to files in a standard file manager.

Units are created by *dragging* them from the ToolBox window to the desired position in the MainGrid window and then connected together by dragging from an output socket on a sending unit to an input socket of the receiving unit. The algorithm is run by clicking on the *start* button (see Figure 2), in a *single step* fashion (i.e., one step at a time) or continuously.

3. New Features and Extensions

Groups of units can now be saved along with their respective parameters. Such groups can also contain groups, which can contain other groups and so on. This is a very powerful feature which allows the programmer to hide the complexity of programs and use groups as if they were simply units themselves. Many improvements have been made to the graphical interface, including compacting the look and style of the toolbox, adding a snap-to cable layout and many more informative windows. The major change however, is that now Grid consists of an object connecting language (OCL) and a separate user interface. This means that the units can be run from within the user interface or as a stand-alone program.

Collaborators are working on tools for various signal and image problems, multimedia teaching aids and even to construct a musical composition system. Currently, in our new release we have toolboxes for various signal processing procedures, animation and a number of image processing/manipulation routines, text processing tools e.g., find and replace, grep and line counting recursively through subdirectories, mathematical and statistical units, a general purpose mathematical calculator (see next section) and a number of flexible importing and exporting units.

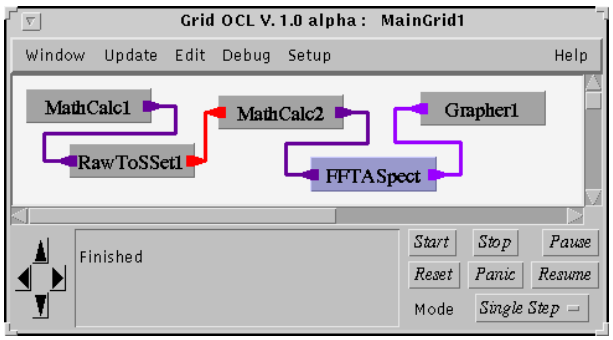

Figure 2. A snapshot of Grid OCL's programming window.

4. MathCalc

The MathCalc unit interprets, optimises and evaluates arithmetic expressions using stream-oriented arithmetic. It recognises a large number of functions and constants. It can be used to evaluate scalar expressions, to process input data sets, or to generate output data sets. All calculations are performed in double-precision real arithmetic.

Stream-oriented arithmetic can be defined as the application of an arithmetic expression to each element of a stream independently. Thus, if B is the sequence $b1, b2, .., bn$, then the function $sin(B)$ evaluates to the sequence $sin(b1), sin(b2), ..., sin(bn)$. MathCalc distinguishes between *constants* and *sequences* or sets. Sets (data sets) are sequences of numbers and constants are single numbers, essentially sequences of length 1. In a MathCalc expression the two can be mixed very freely, with the restriction that all sequences must have the same length. Sequences or constants can be obtained from the input nodes of the MathCalc unit. The example given in the MainGrid window (see Figure 2) demonstrates the flexibility of the MathCalc unit.

The first *MathCalc* unit creates a 125 Hz sine wave by using the equation $sin(((sequence(512) * 2) * PI) * 0.125)$ where the sample rate is 1kHz (MathCalc will optimise this to (2*PI*0.125) * sequence(512)). This is then transformed into a SampleSet type by adding its sampling frequency (i.e., 1 kHz). The second MathCalc unit adds Gaussian noise to its input (i.e., by typing $gaussian(512)+\#0s$). The #0 means node 0 and the s means that it is a sequence as opposed to a c which would a constant. The resultant amplitude spectrum (FFTASpect) is shown from Grapher1 (see Figure 3).

Once the signal is displayed, it can be investigated further by using one of the Grapher's various zooming facilities. Zooming can be controlled via a zoom window which allows specific ranges to be set or by simply using the mouse to *drag* throughout the image. For example, by holding the control key down and dragging down the image is zoomed in vertically and by dragging across from left to right zoomed in horizontally. The reverse operations allow zooming out. Also once zoomed in, by holding the shift key and the control key down the mouse can be used to move around the particular area you are interested in. We also have another powerful zooming function which literally allows the user to drag to the position of interest and the image will zoom in accordingly.

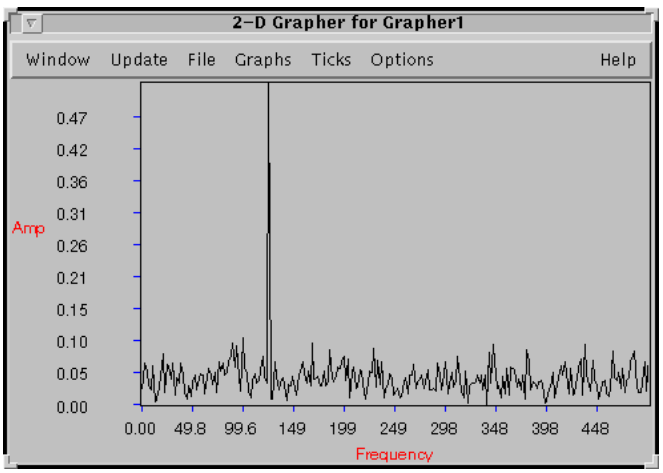

Figure 3. Grapher1's output: any grapher can simultaneously display many signals each with its own colour and line style.

5. Grid OCL : Past and Present

Grid originated from an implementation of the system using C++ and InterViews (Taylor & Schutz 1995) but was abandoned in early 1996. Version two (Taylor & Schutz 1996) was written using the Java Development Kit, JDK 1.0.2 but this was updated in order to be compatible with the new JDK 1.1.x kit. We also re-implemented the base classes and to create OCL. The most recent version of Grid OCL (in November 1997) is a late *alpha* and goes by a different name (*Triana OCL*[4]). Triana OCL will be entering its beta testing stage early next year followed by a final version shortly after. We are in the process of being able to provide a commercial version of the software for which support can be given. None-the-less we will always provide it in a free downloadable from the WWW with a certain time limit (3 or 4 months). Our main goal is to create a very wide user base.

References

Taylor, I. J. & Schutz, B. F. 1995, The Grid Musical-Signal Processing System, International Computer Music Conference, 371

Taylor, I. J. & Schutz, B. F. 1996, The Grid Signal Processing System. in ASP Conf. Ser., Vol. 125, Astronomical Data Analysis Software and Systems VI, ed. Gareth Hunt & H. E. Payne (San Francisco: ASP), 18

[4]http://www.astro.cf.ac.uk/Triana/