METEOROLOGICAL DATA VISUALIZATION USING IBM VISUALIZATION DATA EXPLORER

Dr. David Watson
IBM UK Laboratories
Winchester, UK

1. INTRODUCTION

IBM Visualization Data Explorer is a general purpose toolkit for the display of data which has been derived, or collected. Data Explorer uses visual programming (Figure 1) to link together modules which perform individual tasks. These modules may be from the supplied set or user written using the Data Explorer library of functions.

In this paper I describe some of the general features of Data Explorer such as the underlying data model, the product's client-server structure, and the provision of parallel execution modes. The remainder of the discussion covers some examples of the use of Data Explorer on data provided by ECMWF.

It is important to note that Data Explorer, although developed and marketed by IBM is also available on workstations from SUN, Hewlett-Packard, DEC, SGI, and Data General, in addition to IBM. On selected platforms from SUN, SGI, and IBM, a symmetric multiprocessor version is available, this is discussed in greater depth within the text.

2. DATA MODEL

2.1 **The reasons for a data model**

When designing a software product the requirements for effective data handling are paramount. However, there is a difference in approach to software design when one considers a bespoke piece of code for a well defined purpose and a generalised system for wide ranging use. When a custom product is to be developed the data formats may be specific for that problem which makes the design somewhat less complex. For a generalised visualization toolkit one needs to have the capability to import a wide variety of data. In designing Data Explorer the underlying structure of data was examined in depth to build an abstract definition of the problem. The approach taken is documented in Haber, Lucas & Collins (1991), briefly, the fundamental core of the model is that data can be represented using a few basic components. For spatial data the important components are:

- positions in space
- data values
- connections between positions.

With these three components one is able to describe most data which exists in the spatial domain. One can also assign data parameters to one of the cartesian axes when non-spatial data is under examination.

An effective data model makes data definition to the system, and operation of the software, easier and more flexible than in a system where the data structures are not coherent. Building on the strengths of the underlying data model gives Data Explorer some notable features of relevance to atmospheric work.

2.2 Notable features of the data model

A common problem in many visualization systems is that data must exist or be coerced onto a regular quadrilateral or hexahedral grid. The data model used by Data Explorer does not impose such restrictions. Data can exist in any arrangement of connection element types, the positions can be arranged irregularly with connection element types being of various sizes. This feature of the Data Explorer data model allows atmospheric data, with variable spacing in the vertical axis, to be handled simply. In this particular case Data Explorer makes definition of the grid structure straight-forward by allowing the user to simply define the grid on which the data has been collected or simulated (for example over the earth's surface at the 1000 hPa level) and provide a single line of the vertical spacing. The complete grid will then be built internally by Data Explorer.

By allowing a number of parameters to share a common grid Data Explorer reduces the total amount of data required to be defined to the system. For example, temperature, pressure, and cloud cover from a simulation can all be defined using one grid and set of connections. Users can then simply switch between data parameters to compare and contrast results. Any number of parameters can be associated with a single grid.

Grids which are related can also be grouped together to allow them to be handled as entities. This allows, for example, all the data from a number of small scale simulations which are on different parts of the globe to held together for the purposes of visualization. An extension of this idea is that overlapping or adjoining grids, of possibly different densities, can be grouped into what is known as a multigrid. A multigrid treats a group of grids as if it where one grid. Realization functions such as streamlines or isosurface (contours) will produce visualizations which ignore the multigrid internal boundaries where appropriate.

For a visualization system to be complete it must have facilities to handle time series of data. Most disciplines in science today generate data which varies over time. Data Explorer allows the definition of times series of data which are either snapshots of data on a single grid, or data where both the parameter values and the underlying grid structure change over time. This gives complete flexibility to users in examining their time series data.

The data model underlies all of the functions of Data Explorer. Detailed understanding of the model allows users to exploit it's power to the full. However, users need only a basic knowledge to achieve most of what is possible.

3. VISUAL PROGRAMMING

3.1 What is visual programming ?

Data Explorer visual programming allows users to combine a collection of supplied and user written tools together in arbitrary ways to create a visual representation of a program to operate on input data. Figure 1 shows an example Data Explorer visual program showing a collection of tools joined together by wires which can be thought of as data paths. In some systems similar to Data Explorer data actually flows along these wires, this process gave rise to the name data-flow to describe such visual programming systems. Data Explorer is, however, a data reference structure where only pointers to data pass between tools. This minimises the use of memory to only that required rather than sending many copies of data between different modules.

3.2 Advantages of visual programming

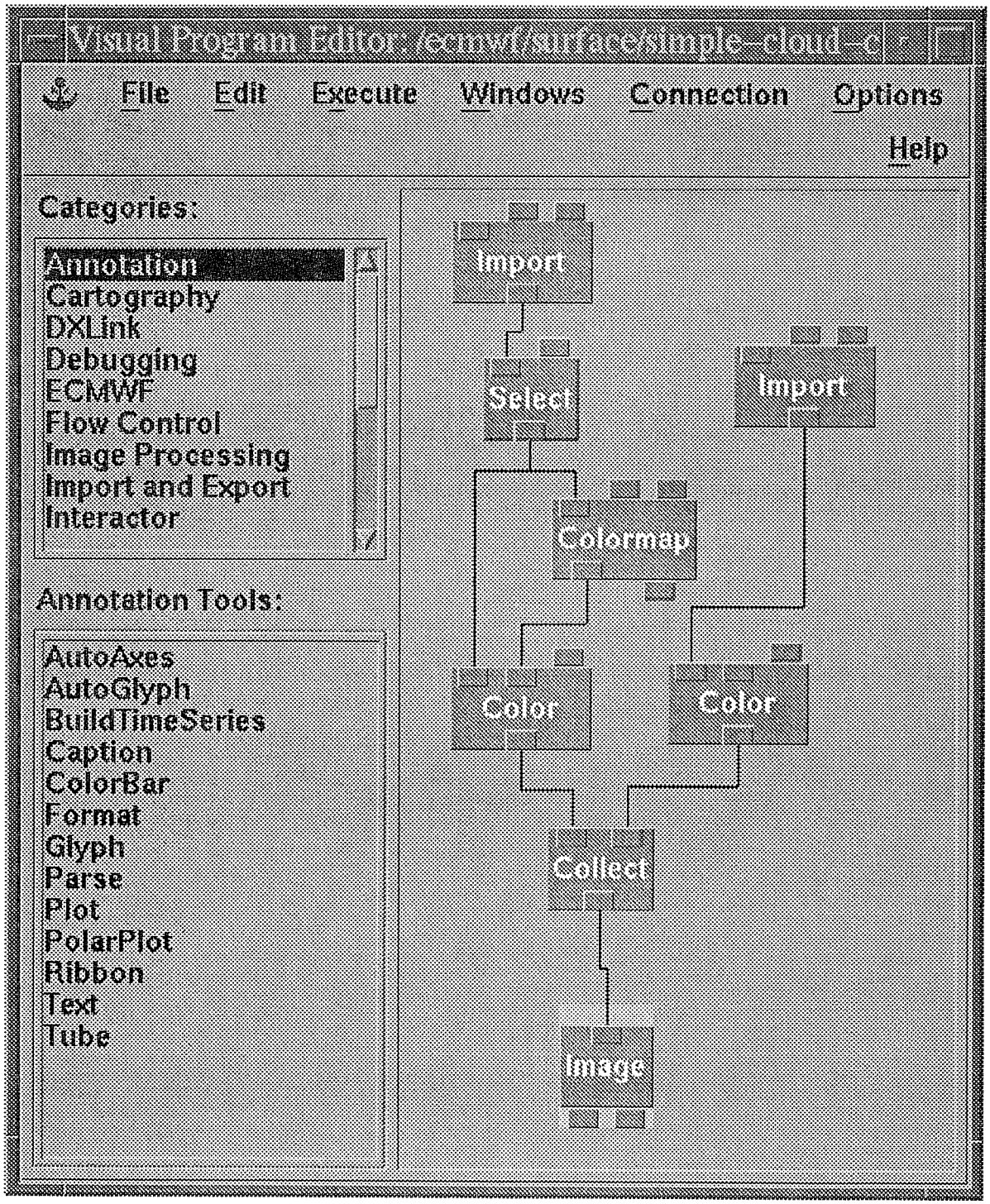
Visual programming has many advantages over traditional methods. Users require a much lower level of skill to perform operations using the supplied tools in a visual programming environment than to build a system in a traditional manner. Flexibility of choice in how it is built and what the visual program does means that it is more complex than a turn-key system. However, users can configure a visual program to meet their exact requirements rather than use the system in the way the developers of a turn-key solution demand.

A common feature of visual programming environments for data visualization is the ability to extend the system with user supplied routines. This allows users to extend and tailor the system to their own requirements. Data Explorer can have three types of user supplied modules added to it:

- inboard modules: those which are used to build a new copy of the Data Explorer executive process
- outboard modules: not link edited with Data Explorer's executive process
- callable modules: those which are part of a shared library

Overall visual programming can be significantly faster for the development of applications and maintenance than traditional programming methods. The visual program is largely self documenting in addition to user supplied help and comment information which can be stored by Data Explorer to aid the user and developer.

FIGURE 1 EXAMPLE DATA EXPLORER VISUAL PROGRAM



4. EXECUTION MODES

4.1 Client / Server

The main Data Explorer application consists of two UNIX processes: dxui and dxexec. The dxui process handles all user interaction with the system, the Visual Program Editor, user defined Control Panels, and Image windows, this is the client process. Processing of the data to create the required visualization is by the dxexec process, this is the server. Communication between client and server is via TCP/IP sockets.

One of the major advantages of the client / server architecture of Data Explorer is the ability to run the two processes on different machines, even when the two machines are from different hardware manufacturers. For example, the user interface may be run on a relatively low powered machine on the desktop whilst the compute intensive process handling the data is run on a powerful server available to a group of users. Data Explorer is available to run on machines from: IBM (including SP/2), Silicon Graphics Inc., Hewlett-Packard, Sun Microsystems, Digital Equipment Corp., and Data General.

4.2 Parallel Execution

In many areas of science the volumes of data being generated has reached unprecedented levels. To process such large amounts of data requires parallel execution of tasks for reasonable performance. Data Explorer is one of very few commercial visualization systems which was designed from the outset to run in parallel.

Generally two types of parallel systems can be described, a distributed system and a shared memory system. Software can be written to exploit these architectures in a number of ways. A basic decision to be made by the designer is whether to decompose the data into smaller areas or split up the operations to be performed into threads which can be executed in parallel. Splitting the data means that one operation is performed on each small area in parallel, bringing the results together before passing the data to the next module. Data Explorer supports both types of parallel execution on suitable hardware configurations.

To achieve maximum performance from parallel code one must reduce the amount of data and message transmission between computational nodes to a minimum. Therefore when a data decomposition method is to be exploited a shared memory, multiprocessor machine configuration is desirable. At present Data Explorer support such Symmetric Multi Processor (SMP) machines from IBM, Sun Microsystems, and Silicon Graphics Inc..

Users may also spread the amount of work to be undertaken across a network of, possibly heterogeneous, machines to achieve parallel execution of the threads in a visual program. Care should be taken to minimise

data transfer in order to achieve maximum performance. If one of the nodes to which an execution group of modules has been assigned is an SMP machine then further parallelism is possible via data decomposition.

Data Explorer offers a complete spectrum of parallel execution choice to the user, this enables complex or large datasets to be handled more effectively than is possible with systems which offer fewer facilities for parallelism.

5. VISUALIZATION OF ECMWF DATA

In this section I describe a selection of visualizations produced using data supplied by ECMWF specifically for the purpose of this paper. The visualizations were all produced using standard supplied tools in Data Explorer.

5.1 Description of supplied data

Two sets of data were supplied, 30 time steps over the whole globe, with surface data and upper air data. The horizontal resolution of both datasets was 2.5 degrees in both latitude and longitude. The surface data was for:

- large scale precipitation
- convective precipitation
- total cloud cover
- 2 metre temperature.

In the upper air dataset the following parameters where given at 1000, 925, 850, 700, 500, 400, 300, 250, 200, 150, 100, 70, 50, 30, & 10 hPa:

- geopotential
- temperature
- u-velocity
- v-velocity
- vertical velocity
- relative humidity.

The data was supplied in GRIB format with the appropriate routines for unpacking the files.

5.2 Describing the data to Data Explorer

Data Explorer has two data description languages supplied with the product, one is the external representation of the internal format used within the product. The other is called the General Array Importer, this is particularly useful for non-programmers. A supplied tool called the Data prompter can be used to describe data to the system. Data Prompter is a menu driven tool which allows users to browse and describe their data to Data Explorer, a General Array Importer header file is built to define the dataset.

For the purposes of this exercise I used the native Data Explorer data format as it is best suited to the type of data supplied.

A set of 'C' routines were used to extract the data from ASCII files created using the supplied routines. Each parameter time step was placed in a separate binary file and the Data Explorer data description file was generated by the same routine.
In example 1 the file describing large scale precipitation in the surface data is shown (suitably reduced in size for clarity). The grid is described by the gridpositions class with user controlled deltas, this is followed by a definition of the grid connections, which in this case are quads. Each time step is then described in turn, the data type such as float, the rank such (0=scalar data), the number of points, and where the data is (in a file called p142_tx starting at byte 0, where x is the time step). After each time step is defined (in a way which enables it to be accessed by name within a Data Explorer program), and object called "142" is defined which is the time series of the Large Scale Precipitation data. Each time step is given a position which can be accessed from within the visual program.

Example 1 Data Explorer description of Large Scale Precipitation Surface Data

```
object 1 class gridpositions counts 73 144
origin -180 90
delta  0    -2.5
delta  2.517  0

object 2 class gridconnections counts 73 144
attribute "element type" string "quads"attribute "ref" string "positions"

object 6 class array type float rank 0 items 10512 msb ieee data file ./p142_t6,0
attribute "dep" string "positions"

object "Time 6" class field
component "positions" value 1
component "connections" value 2
component "data" value 6

object 12 class array type float rank 0 items 10512 msb ieee data file ./p142_t12,0
attribute "dep" string "positions"

object "Time 12" class field
component "positions" value 1
component "connections" value 2
component "data" value 12

..... time steps 18 to 216 omitted for clarity .....

object "Time 216" class field
```

```
component "positions" value 1
component "connections" value 2
component "data" value 216

object 228 class array type float rank 0 items 10512 msb ieee data file ./p142_t228,0
attribute "dep" string "positions"

object "Time 228" class field
component "positions" value 1
component "connections" value 2
component "data" value 228

object 240 class array type float rank 0 items 10512 msb ieee data file ./p142_t240,0
attribute "dep" string "positions"

object "Time 240" class field
component "positions" value 1
component "connections" value 2
component "data" value 240

object "142" class series
member 0 position 6 "Time 6"
member 1 position 12 "Time 12"

..... time steps 18 to 216 omitted for clarity .....

member 28 position 228 "Time 228"
member 29 position 240 "Time 240"
```

For the Upper Air data the major difference in definition is the way in which the grid is described to Data Explorer. In example 2 the file (again reduced in size) for Geopotential is shown. A base grid is defined as object 1000 which has a third dimension of one. Object 1001 is a line in three dimensions where only the third dimension varies. Points on the third dimension are at the given pressure levels for the Upper Air data. Using Data Explorer's productarray class the base grid and the vertical pressure line are combined to produce a three dimensional grid which is then used in the description of each time step in the say was as for the Surface data shown above.

Example 2 Data Explorer description of Geopotential data in the Upper Air dataset

```
object 1000 class gridpositions counts 73 144 1
origin -180 90 0
delta  0  -2.5 0
delta  2.5  0 0

object 1001 class array type float rank 1 shape 3 items 15 data follows
0.0 0.0 1000.0 0.0 0.0 925.0 0.0 0.0 850.0 0.0 0.0 700.0 0.0 0.0 500.0 0.0 0.0 400.0 0.0 0.0 300.0
```

0.0 0.0 250.0 0.0 0.0 200.0 0.0 0.0 150.0 0.0 0.0 100.0 0.0 0.0 70.0 0.0 0.0 50.0 0.0 0.0 30.0 0.0 0.0 10.0

object 1 class productarray
term 1001
term 1000

object 2 class gridconnections counts 15 73 144
attribute "element type" string "cubes"
attribute "ref" string "positions"

object 6 class array type float rank 0 items 157680 msb ieee data file ./p129_t6,0
attribute "dep" string "positions"

object "Time 6" class field
component "positions" value 1
component "connections" value 2
component "data" value 6

object 12 class array type float rank 0 items 157680 msb ieee data file ./p129_t12,0
attribute "dep" string "positions"

object "Time 12" class field
component "positions" value 1
component "connections" value 2
component "data" value 12

..... time steps 18 to 216 omiited for clarity .....

object "Time 228" class field
component "positions" value 1
component "connections" value 2
component "data" value 228

object 240 class array type float rank 0 items 157680 msb ieee data file ./p129_t240,0
attribute "dep" string "positions"

object "Time 240" class field
component "positions" value 1
component "connections" value 2
component "data" value 240

object "129" class series
member 0 position 6 "Time 6"
member 1 position 12 "Time 12"
member 2 position 18 "Time 18"

..... time steps 18 to 216 omiited for clarity .....
....
member 28 position 228 "Time 228"
member 29 position 240 "Time 240"

A further alternative to bring data into Data Explorer is for the user to write a custom data reader and incorporate it into the system. This is the preferred way when numerous files of the same type are to be read regularly. If a long-term meteorological project were to be undertaken a GRIB reader should be written.

5.3 Surface plot

Figure 2 shows a 2-D surface plot of cloud cover from the surface dataset. Only cloud cover above 80% is shown in this image. The world political boundary map has been included for reference, this is simply another Data Explorer file which is combined with the cloud data in a simply operation in the visual program. Although shown here in greyscale the image on the screen would generally be colour under the control of a user defined colourmap.

5.4 Visualization through the atmosphere

In Figure 3 three slices have been taken from an irregularly spaced (in the vertical direction) 3-D dataset of the upper air. Political boundaries are shown on the lowest pressure level. Choice of the number and position of the slices is arbitrary and under user control. Within the Data Explorer program the temperature data given in degrees Kelvin has been converted to Celsius using the Compute tool. Compute is a mathematical interpreter included in Data Explorer for the manipulation of data within the visual program. It supports all of the 'C' language math.h built in functions such as sin, and log together with vector operations such as dot product or vector element selection.

5.5 Deformation to a globe and other map projections

Figure 4 shows an example of manipulation of a component of the upper air data by using the Compute module to warp the data to a globe. The positions component of the upper air data is repositioned to form a sphere, all of the 3D positions are changed such that lower pressure levels are placed further away from the surface. The landmasses are produced by taking the earth topography dataset and defining a colourmap with a solid black for above sea-level and a light grey for below sea-level.

Windforce contours have been produced by first combining the u and v velocity components and taking the magnitude of the resulting vector (all done using the Compute module), then using the Isosurface module to produce contours. As the data is a time series the whole, or part, of the series can be sequenced through using Data Explorer supplied tools, to build an animation of the data in a spherical representation as easily as in 2D.

FIGURE 2 CLOUD COVER FROM THE SURFACE DATASET

FIGURE 3 TEMPERATURE SHOWN AT 0, 500, & 1000 hPa

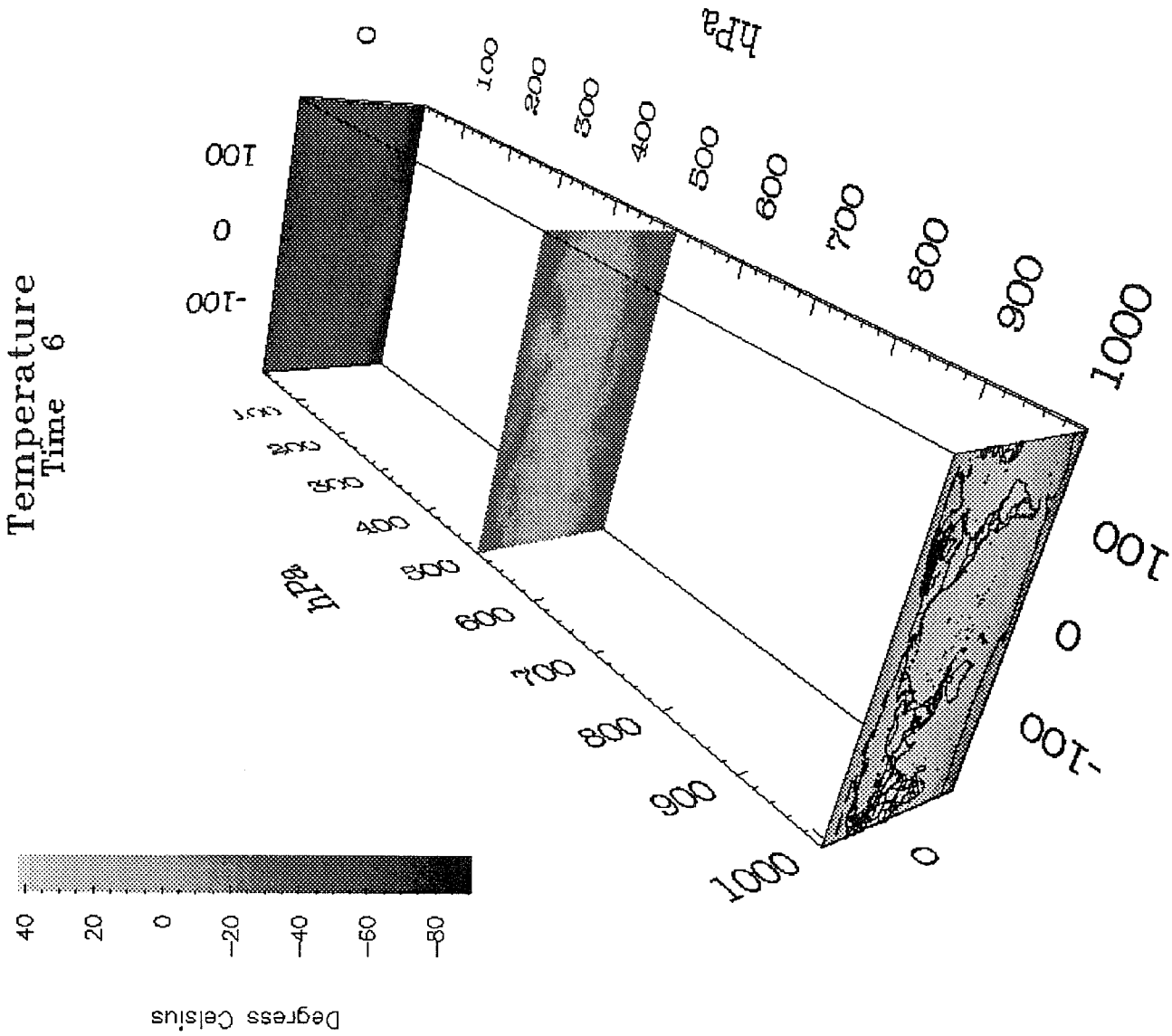


FIGURE 4 WINDFORCE SHOWN ON A CYLINDRIC PROJECTION GLOBE

Using Compute users can transform data into any cartographic projection such as Mollweide, sinusoidal, Goode, or Eckert. The example in Figure 3 is a simple cylindric projection. Few systems allow such flexibility in the control of data in this way by the user.

6. CONCLUSIONS

IBM Visualization Data Explorer is a full-function, visual programming environment for data visualization. Data from any source can be described to the comprehensive unified data model. Users can develop quickly, programs to explore and represent their data. Flexibility in the system allows for rapid prototyping, maintenance, and addition of user supplied tools.

Data supplied by ECMWF was defined to the system and a number of representative data visualizations were developed for this paper. No user code was added to the system, only supplied tools were used in development of the visualizations. The system has been shown to be capable of handling meteorological data in particular the irregular spacing of pressure levels through the atmosphere and time series. Data manipulation and cartographic transformations can be performed with ease using Data Explorer.

7 REFERENCES

Haber, R.B., Lucas, B., Collins, N., 1991, A Data Model for Scientific Visualization with Provisions for Regular and Irregular Grids. Proceedings, IEEE Visualization '91, San Diego. 298-305.

8 DATA

ECMWF 1995. The Description of the ECMWF/WCRP Level III-A Global Atmospheric Data Archive