Data Explorer

Background and history
OpenDX
Example: Laplace data set
Reading

- IBM Visualization Data Explorer QuickStart Guide
  - most likely out of date (1997) but covers main ideas, modules

- On-line tutorials
  - OpenDX tutorials on the p655 (TBD)
  - Dept of Oceanography at Dalhousie (Nova Scotia)
    - http://www.phys.ocean.dal.ca/docs/DX_tutorial.html
Gallery

- The next few slides have some snapshots to show what’s possible
  - this view is basically a histogram, using 3D rendering
  - perspectives, shading help distinguish categories
Gallery (cont’d)

- This is a frame from an MPEG movie produced by DX
  - uses a transparency control to draw the outer layers
  - see fMRI-MRI.mpg
Gallery (cont’d)

- An image from the Dalhousie tutorial
Gallery (cont’d)

- CIS 455/555 term project (1995) using Iris Explorer

magma chamber below the East Pacific Rise
History

- Data Explorer was an IBM product

- One of many systems based on the notion of a *visualization pipeline*
  - Others: AVS, Khoros, SGI’s Iris Explorer

- Visualizations are the result of several transformations that can be independently applied

- Use a “dataflow” model to move data through a series of operations
OpenDX

- In 1999 IBM released the source code
- OpenDX is the open source project that maintains, extends DX
  - currently at version 4.3.2 (Nov 2003)
- Download, compile for any system that has X Windows
  - uses Motif and OpenGL
- See link to company that sells Mac OS/X port
OpenDX on the p655

- Set these environment variables in your .login:
  
  ```bash
  setenv DXROOT /usr/local/share/opendx/dx
  setenv DXMACROS /usr/local/share/opendx/dx/macros
  setenv DATA /usr/local/share/opendx/dx/data
  ```

- Update your path:

  ```bash
  setenv PATH /usr/local/share/opendx/bin:$PATH
  ```

- Start DX:

  ```bash
  % dx &
  ```
Caveats

- This is a very complex system
- Plan on investing some time to learn it
  - as always, working through tutorials is a good way to start
- The motif interface is very outdated
- Documentation not the best, has inconsistencies
- But the results are very nice (nicer than the screen-shots on the previous slides)
Philosophy

- A major theme in DX: users should control the presentation
  - static displays defined by the application that produces the data are not as effective as environments that allow users to interact with the data

- The “explorer” part of the DX name implies users can
  - select which part of the data to view
  - modify the presentation (e.g. change colormaps)
  - interact with the data, e.g. by changing perspective, view angle, etc

- Example: scatter plots
  - application produces data with a large number of attributes
  - user selects pairs of attributes, glyphs, orientation, axes, etc
  - can change selections as they view the data
Visual Programming

- A visualization in DX is defined by a program
  - read input data
  - apply functions
  - output is a rendering

- The sequence of operations can be stored in a file
  - edit, update for future visualizations

- The language used to describe the pipeline is a visual programming language (VPL)

- VPLs have been used in a wide variety of application areas
  - See M. Burnett’s research on Forms/3 at OSU
  - http://web.engr.oregonstate.edu/~burnett/Forms3/forms3.html
Example

- When you are working on a program the “code” appears on a canvas
- Objects that can be added to the program are in a palette on the left side of the window
- Click on an object, then click in the program to place an instance
- Connections between tabs on objects show flow of data between the objects
Example (cont’d)

- To connect two objects
  - click an output tab on the source object
  - click an input tab on the destination
- The system draws connecting arcs automatically
- In this program:
  - a FileSelector pops up a file browser, lets the user select an input
  - the Import object reads the data, converts it to an internal format
  - different types of Import objects interface to various data formats
Aside: Scientific Data Formats

- Scientific data can be stored in a variety of ways
- Several groups have defined formats in an attempt to create a standard representation
  - HDF (hierarchical data format)
    - [http://hdf.ncsa.uiuc.edu/](http://hdf.ncsa.uiuc.edu/)
  - CDF (common data format)
  - many others
- These formats combine *meta-data* records with the data itself
  - meta-data describes the data
  - number of dimensions, type (*float* vs *int*), creation date, many other attributes
Data Formats and DX

- OpenDX will import data in a wide variety of formats, including:
  - HDF
  - CDF
  - Images (e.g. gif and jpeg)
  - Spreadsheet

- A simple and very useful format that helps import data produced by other applications (e.g. our Laplace programs) is the general array format:
  - Create a single text file with the meta-data
  - A field in this file will be the name of the file containing the data itself
Meta-Data for the Laplace Output

- Run the laplace program to make the main data file:
  
  ```
  % laplace -nx 200 -ny 200 > laplace.txt
  ```

- Create a meta-data file that describes the data:

  ```
  file = /home/users/conery/classes/455/dx/laplace.txt
  grid = 200 x 200
  format = ascii
  type = float
  etc
  ```

```plaintext
100 100 100...
0 49.9 69.6.. 
0 30.1 49.8..
...
```
Meta-Data for the Laplace Output (cont’d)

- Note you only have to make the meta-data file once
- You can update the main data file independently
  - e.g. edit, test the Laplace application
  - run again to make a new data
- The meta-data file is the file loaded into DX
  - “laplace.general” is entered as the file name in the browser

```plaintext
file = laplace.txt
grid = 200 x 200
...
```

```plaintext
100 100 100 ...
0 49.9 69.6..
0 30.1 49.8..
...
```
Use DX to Define the Meta-Data File

- DX has a “wizard” that will help you make the meta-data file
  
  `dx -prompter`

- Select “grid or general array format”
Prompter (cont’d)

- Clicking the button opens up a panel
- Select the main options (the defaults are the right choice for the Laplace data)
- Click “describe data”
Prompter (cont’d)

- A new window pops up -- enter data descriptions here, then save
Test the New File

- After you save the file and close the format definition window, three buttons in the prompter window will be enabled
  - **Test Import...**
    - launches a DX program that has an import module
    - reports statistics about your file (e.g. number of cells, dimensions, ...)

![Screenshot of prompter window with options for data organization and buttons for actions]
Test the New File (cont’d)

- The **Visualize Data**... button creates a new DX program based on your data type
  - use this program as a template
  - save it in a file, edit it later to add new controls, etc

- The new program is run, and it displays the default view
  - 3D image, viewed from “the top” (see next slide)
  - control to change orientation (location of camera in 3D space)
Output from the Default Viewer
Details

- The default view for 2D data is a “rubber sheet”
  - a surface is displayed above the x-y plane
  - elevation at \((i, j)\) is defined by the data value at \(A[i, j]\)

- Note the surface is smooth, not a “mesh”
  - shading is defined by computing normals at each point
  - (these were also used to give the sea bottom a more realistic appearance)
View After Rotating
Look at Your DX Program

- Click on “open visual program editor” under the windows menu to open your program
- It’s a large program, spread over many panels
- Click tabs at the top of the page to see different “modules”
Quick Survey of DX Functions

- The **Include** module can select a subset of its input data.
- **Colormap** provides a very flexible method for coloring data according to its value (e.g., in an image):
  - double-click on the **Colormap** module to bring up a control panel;
  - change the mapping by RGB, HSV, etc.
Survey (cont’d)

- The ColorBar generates a scale to show the mapping.
- Collect combines several objects into a single object.
  - In this example the ColorBar overlays the colored image.
Survey (cont’d)

- The RubberSheet module produces the 3D view of 2D data
- Normals supplies shading (see next slide)
Survey (cont’d)

RubberSheet

RubberSheet plus Normals
Survey (cont’d)

- To break a program into smaller pieces, define *transmitters* and *receivers*
  - names are italicized in the module display
Survey (cont’d)

- See the Dalhousie tutorial for examples of other modules
  - methods for displaying text (labels, titles, etc)
  - vector fields
  - frames for animation
Scripts

- It is possible to have DX controlled by another program
- Use the DXLink API to launch DX, send it commands, access the results
  - Py-OpenDX for Python
  - Perl??
- The DXCallModule API is similar, but invokes just one module
  - with DXLink you pass DX a program name, and it calls all the modules in the program
- DX is also extensibile
  - write new modules that can be dynamically linked