Introduction to OpenDX

Mike Bailey

Oregon State University
OpenDX

- Started out life as *IBM Visualization Data Explorer*
- When the product was cancelled, IBM put it into Open Source and renamed it *OpenDX*
- Basic premise is a series of interconnected modules, living together in an environment called the Visual Program Editor (VPE)
- There are lots of provided modules
- You can also write your own

http://www.opendx.org
http://www.vizsolutions.com
http://eecs.oregonstate.edu/~mjb/opendx
The Structure of an OpenDX Module

Inputs

OpenDX Module

Code that does something useful

Outputs
Steps in Creating a Visualization
Seven Steps to Creating a Visualization

1. Get the data
2. Formulate a scientific strategy. What do you want to show? How do you want to show it?
3. Import the data
4. Create a *simple* OpenDX network
5. Incrementally embellish the network. Save it often!
6. Choose what quantities you want to interact with. Change the Interactor styles to match the quantities being modified.
7. Create the output.
Starting OpenDX in OSU’s Computer Graphics Education Lab
Starting OpenDX on the OSU CGEL Systems

1. Start → All Programs → WRQ Reflection → Reflection X
2. In the View menu, click off X Desktop
3. Minimize the Reflection X window (the _ in the upper right corner)
4. Start → All Programs → OpenDX → DX

Quitting OpenDX on the OSU CGEL Systems

1. Select Quit from the OpenDX Main Menu
2. Maximize the Reflection X window by clicking here in the Task Bar
3. In the File menu, select Exit
The OpenDX Main Menu and Categories of Modules
The OpenDX Main Menu

- Get into the Data Prompter program
- Run an OpenDX network without seeing the network
- Run an OpenDX network and be able to edit the network
- Run the internal OpenDX tutorial
- Create a new OpenDX network
- Load, and be able to edit, one of the OpenDX sample networks
- Exit OpenDX
Nine Categories of OpenDX Modules

- Annotation
- Interactor
- Special
- Debugging
- Realization
- Structuring
- Import & Export
- Rendering
- Transformation
Annotation OpenDX Modules

- AutoAxes – creates an axis box for whatever data you are plotting
- AutoGlyph – designs and produces glyphs for the data based on the data values
- Caption – creates caption text for an image
- ColorBar -- creates a colorbar to be displayed
- Format – creates a string from a number (used to create file names)
- Glyph – produces an identical glyph for every point in the data
- Legend – produces a legend to be displayed
- Plot – creates a 2D plot
- Ribbon – creates a flow field ribbon
- Text – displays text in 3D space
- Tube – creates a flowfield tube
Debugging OpenDX Modules

- Describe – describes an object
- Print – prints information about a field to the Message Window
Import & Export OpenDX Modules

- Export – writes data from OpenDX into a file
- Import – reads data into OpenDX from a file
- ImportSpreadsheet – reads data into OpenDX from a tabular file
- Include – includes or excludes points in a field based on their data values
- ReadImage – reads an image into OpenDX from a file
- Reduce – filters and resamples a field into a lower resolution
- Refine – interpolates a field into a higher resolution
- Slab – takes a positional subset of the data
- Slice – takes a positional slice through the data
- WriteImage – writes an image from OpenDX into a file
Interactor OpenDX Modules

• FileSelector – presents a dialog box to let you select a file
• Integer – allows the user to input an integer number
• Scalar – allows the user to input a floating point number
• Selector – allows the user to select one of a number of options
• String – allows the user to input a string
• Toggle – allows the user to select one of two options
• Vector – allows the user to input a vector
Realization OpenDX Modules

- AutoGrid – maps a set of scattered points onto a grid
- Band – divides a field into bands
- Connect – creates triangle connections for scattered data points in a field
- IsoSurface – creates surfaces or lines of constant data value
- MapToPlane – projects a data field onto an arbitrary plane
- RubberSheet – deforms a surface field by the amount of the data value at each point
- ShowBox – creates a bounding box for display
- ShowConnections – displays the outline of connectivity elements in a field
- ShowPositions – displays the positions in a field
- Streakline – computes an advection path through a changing flow field
- Streamline – computes a path through a non-changing flow field
Rendering OpenDX Modules

- AmbientLight – specifies the ambient light
- Arrange – creates a single side-by-side image from a collection of images
- AutoCamera – selects a good camera view of the data
- Camera – specifies a camera view
- Display – a more elaborate image-rendering system than Image
- Image – renders and displays field data
- Light – specifies a distant (parallel) light source
- Normals – compute point or face normals for shading a surface
- Render – renders a field and creates an image
- Rotate – rotates field data
- Scale – scales field data
- Shade – specifies object-shading parameters
- Transform – performs a general matrix transform of an object
- Translate – translates field data
Special OpenDX Modules

Colormap – presents an interactive tool for specifying color vs. data value
Receiver – receives the output of a Transmitter
Sequencer – creates an animation “VCR” display
Transmitter – “wirelessly” connects a network to a receiver
Structuring OpenDX Modules

- Collect – collects objects into a group
- Inquire – returns information about a field
- Mark – marks a new field component as “data” (e.g., for Compute)
- Remove – removes a specified component from a field
- Rename – renames a specified component in a field
- Unmark – undoes the effects of Mark
Transformation OpenDX Modules

- AutoColor – automatically color a data field (blue → green → red)
- Color – assign a color by name of by RGB values
- Compute – perform point-by-point arithmetic on a field’s “data” component
- DivCurl – computes the divergence and curl of a flow field
- Equalize – apply histogram equalization to a field
- Gradient – computes the gradient of a scalar field
- Histogram – creates a histogram that can be rendered with Plot
- Map – projects one field’s data onto another field’s geometry
- Measure – calculates surface area and volume of a geometry (e.g., isosurface)
- SimplifySurface – reduces the size of the triangular mesh
- Statistics – computes the mean, standard deviation, variance, minimum, and maximum of a field’s data
Adding and Connecting Modules
Adding a Module into the Visual Editing Area

1. Left-click on the module category to list its modules.
2. Left-click on the module you want to add
3. Move the cursor into the Editing Area and left-click

It's not drag-and-drop, it's click-and-click
Connecting Modules in the Visual Editing Area

1. Left-click on the output tab of the module you are connecting from
2. Keeping the left button down, drag to the input tab of the module you are connecting to
3. When you get close, the tabs to which a connection make sense will highlight in green
4. Move the cursor on top of the tab you want to connect to, and release the left mouse button
5. To disconnect, reverse the process. Click on the input tab and drag back to the output tab.

If an input tab is in the “up” position, you are allowed to try to connect to it.

If an input tab is “down”, then it has already been set to a constant within the module itself, and cannot take an external connection until that constant has been un-set.

Just because an input tab is up, however, doesn't mean that this input is data-compatible with the output you are trying to connect to it. Data-compatibility is indicated by the input tab(s) turning bright green.

This, however, still doesn't imply that the connection makes logical sense. 😊
Some Modules Can Have Variable Numbers of Tabs

Collect and Compute are two common modules that work this way.

Edit → Input/Output Tabs → Add Input Tab
Terrain Visualization
Terrain Visualization

Start simply, then embellish!
The Import and FileSelector Modules

You can type a filename into the Import module, but hooking in a FileSelector module makes it way easier and friendlier.
The first input “tab” is the field input.

- Click on the Hue, Saturation, Value, or Opacity labels to edit that curve.

- Double-click on a line to add a control point there.

- Click on a control point to select it.

- Sweep a box over several control points to select them all.

- Hold down the left mouse button on a control point to move it. If several are currently selected, all will move together.

- Edit → Delete to remove selected control point(s).

The data value range over which the colors apply is determined by scanning the data itself.
The Colormap Editor in Action
Rubbersheeting the Terrain Surface
The Image Window
The **AutoAxes** option has many ways to embellish the visualization with axes, labels, grids, etc.
The **Mode** option lets you set what scene transformation the mouse will perform.
Image Window Options

The `View Control` option lets you set various aspects of how the scene will appear.

<table>
<thead>
<tr>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>View Control</td>
<td>Set various aspects of how the scene will appear.</td>
</tr>
<tr>
<td>Redo</td>
<td>Undo Ctrl+Z</td>
</tr>
<tr>
<td>Ctrl+Y</td>
<td>Redo Ctrl+Y</td>
</tr>
<tr>
<td>Ctrl+F</td>
<td>Reset</td>
</tr>
</tbody>
</table>

- **Same as the Mode option**
- **Set a pre-defined view**
- **Specify Perspective or Orthographic 3D projection**
- **If using Perspective, this specifies the field-of-view angle. The larger this number, the more severe the perspective will be.**

Same as the `Mode` option
Debugging
The Print Module

First argument is the field to print

Second argument is a string with one or more characters:
- \texttt{r} recursively traverse the object
- \texttt{o} print only the top level of the object
- \texttt{d} print first and last 25 items in arrays, as well as headers
- \texttt{D} print all the items in arrays as well as headers
- \texttt{n} print object to \texttt{n} levels.

“\texttt{rD}” works well
Printing to a File ("logging")

Checkbox to turn logging on/off
Specify the file to write to
Scalar Visualization
MapToPlane interpolates the 3D field onto the given plane. The first argument is the field, the second is a 3D point on the plane, and the third argument is a 3D normal to the plane.
3D Cutting Plane – Contours
Direct Volume Rendering

A Volume Rendering “Transfer Function” relates data scalar value to its corresponding color and opacity. For volume rendering, OpenDX uses the color Value as the opacity, not the color Opacity.

The direct volume rendering part of the Image module will only work in Orthographic projection.

These are the “Transfer Function”
Mapping Another Data Field onto Isosurfaces
Vector Visualization
Vector Cloud

Import

AutoGlyph

Colormap

Toggle

Color

Scalar

Switch

Collect

Image

Annotation
Speed Isosurfaces

Compute

Notation: Compute

Inputs:
Name: t
Name: b

Expression:
\[ \sqrt{a.x.a.x+a.y.a.y+a.z.a.z} \]
Streamline Ribbon

Compute

Notation: Compute

Inputs:
Name
a
b

Expression:
sqrt(a.x*a.x + a.y*a.y + a.z*a.z)

Realization
Streamline Tube

Annotation

Expression:
$$\sqrt{a.x^2 + a.y^2 + a.z^2}$$
Curl Transformation
Divergence
Animation
The Compute Module

Does arithmetic on the point-by-point Data component of a field, and outputs the modified field

The 3 (in this case) inputs

The output expression, in this case, a 3-vector with a newly-created Z value
The Compute Module

Does arithmetic on the point-by-point Data component of a field, and outputs the modified field. But, what if you want to do arithmetic on a different component?

The Mark module renames the Data component to something temporary, and renames a component you select to “Data”. Compute then acts on this component.

The Unmark module changes the component names back to what they were originally.

Structuring
Sequencer outputs a series of integers. You set minimum, maximum, and delta using **Edit → Configuration**.
In this case, Compute turns an integer into a scalar to be used to animate an isovalue.
The Sequencer Module: “Percent Units Strategy”

A good Sequencer Strategy: Run the sequence from 1-100 (or 0-100).

Then, base the Compute quantity on these “Percent Units”.

expression: 400.*a/100.
In this case, *Compute* turns an integer into a scalar to be used to animate the isovalue.
The Sequencer Module: Setting a Scalar Isovalue

Cutting plane position = [ 0.0, 0.0, 4.3], Isovalue = 56.0
In this case, **Compute** turns an integer into a 3-element vector to be used to animate the position of the cutting plane.
The Sequencer Module: Setting a Vector to act as a Plane Location
In this case, Compute turns an integer into a rotation angle in degrees.
Rotation and Scaling always occur about the origin. To change this to the center of the volume, translate the volume to the origin, perform the rotation or scale, and then translate it back.

Translate by [-15,-15,-15]

Translate by [15,15,15]
Writing Out a MIFF Animation File
Converting a MIFF Animation File into an Animated GIF File using the ImageMagick Package

Click on: Start → All Programs → Accessories → Command Prompt
Converting a MIFF Animation File into an Animated GIF File using the ImageMagick Package

Type: `convert anim.miff anim.gif`
(where `anim` is the name of your MIFF animation file written from the `Image` module)

http://www.imagemagick.org
Animated GIF Files work in Windows Explorer

Double-click on the animated GIF file
Animated GIF Files work in Web Pages
Animated GIF Files work in PowerPoint

In a presentation, the image will start animating when this slide becomes active.
Interactors
Select an Interactor by *Left-Clicking* its Label.

Then, click *Edit*. You can change the Interactor’s Style, Layout, Attributes, and Label. Under *Set Attributes*, clicking on the *Continuous* checkbox is usually a good thing.
Ganging Interactors

You can place all Interactors in a single window by using the **middle mouse button** to drag them over. This copies them, not moves them. Then select the original Interactor in its original window and *Edit-Delete* it.
Transmitters and Receivers
It’s Easy to Get Cluttered, Especially Around *Import* and *ColorMap*!
It’s Also Easy to Get Un-cluttered with *Transmitter* and *Receiver*

Notice how this lets you create separate “regions” for different functions. Wouldn’t it be nice if you could put each region on its own page?
Using **Transmitter** and **Receiver**, You Can Also Spread the Network Out on Multiple “Pages”

Click *Edit → Page → Create Empty Page* to make a new page

Double-click on the page’s tab, type in the page’s name, and hit Enter

You can create from scratch in the other pages, or cut-and-paste from where you started
Switching, Selecting, and Toggling
Selecting from Multiple Objects: Selector and Switch

The Switch module sends nothing through when its first input is 0. It sends the second input through when the first input is 1. It sends the third input through when the first input is 2, etc.
Toggling Objects On and Off: *Toggle* and *Switch*

The Switch module sends nothing through when its first input is 0. It sends the second input through when the first input is 1. It sends the third input through when the first input is 2, etc.
Captions
Placing a Caption on a Visualization
Setting the Color of the Caption

Click here to get a list of simple colors, or type the name of a color here.
## Color Names you can use in the Color Module

<table>
<thead>
<tr>
<th>Color Name</th>
<th>Color Name</th>
<th>Color Name</th>
<th>Color Name</th>
<th>Color Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>aquamarine</td>
<td>darkturquoise</td>
<td>lightgrey</td>
<td>midnightblue</td>
<td>springgreen</td>
</tr>
<tr>
<td>black</td>
<td>dimgray</td>
<td>lightsteelblue</td>
<td>navy</td>
<td>steelblue</td>
</tr>
<tr>
<td>blue</td>
<td>dimgrey</td>
<td>limegreen</td>
<td>navyblue</td>
<td>tan</td>
</tr>
<tr>
<td>blueviolet</td>
<td>firebrick</td>
<td>magenta</td>
<td>orange</td>
<td>thistle</td>
</tr>
<tr>
<td>brown</td>
<td>forestgreen</td>
<td>maroon</td>
<td>orangered</td>
<td>turquoise</td>
</tr>
<tr>
<td>cadetblue</td>
<td>gold</td>
<td>mediumaquamarine</td>
<td>orchid</td>
<td>violet</td>
</tr>
<tr>
<td>coral</td>
<td>goldenrod</td>
<td>mediumblue</td>
<td>palegreen</td>
<td>violetred</td>
</tr>
<tr>
<td>cornflowerblue</td>
<td>gray</td>
<td>mediumforestgreen</td>
<td>pink</td>
<td>wheat</td>
</tr>
<tr>
<td>cyan</td>
<td>green</td>
<td>mediumgoldenrod</td>
<td>plum</td>
<td>white</td>
</tr>
<tr>
<td>darkgreen</td>
<td>greenyellow</td>
<td>mediumorchid</td>
<td>red</td>
<td>yellow</td>
</tr>
<tr>
<td>darkolivegreen</td>
<td>grey</td>
<td>mediumseagreen</td>
<td>salmon</td>
<td>yellowgreen</td>
</tr>
<tr>
<td>darkorchid</td>
<td>indianred</td>
<td>mediumslateblue</td>
<td>seagreen</td>
<td></td>
</tr>
<tr>
<td>darkslateblue</td>
<td>khaki</td>
<td>mediumspringgreen</td>
<td>sienna</td>
<td></td>
</tr>
<tr>
<td>darkslategray</td>
<td>lightblue</td>
<td>mediumturquoise</td>
<td>skyblue</td>
<td></td>
</tr>
<tr>
<td>darkslategrey</td>
<td>lightgray</td>
<td>mediumvioletred</td>
<td>slateblue</td>
<td></td>
</tr>
</tbody>
</table>
Placing a Data-formatted Caption on a Visualization

The image shows a visualization with a data-formatted caption. The caption is set to display the isovalue as "Isovalue = 40.00."
Importing Your Own Data
OpenDX Data Grid Types

Surfaces

Regular:
R positions,
R connections

Deformed Regular:
IR positions,
R connections

Irregular:
IR positions,
IR connections

Volumes

R = “regular”
IR = “irregular”
Creating an OpenDX Data Descriptor File using the Data Prompter

- Data Explorer file
- CDF format
- NetCDF format file
- HDF format
- Image file
- Grid or Scattered file (General Array Format)
- Grid type
- Number of variables
- Positions in data file
- Single time step
- Data organization:
  - Block
  - Columnar
- Spreadsheet format file

The import module can read DX files.
Most of the time, this is what you want

Regular positions, regular connections

One scalar value at each point

Doesn’t matter here, but a good value for other data

See next page
Enter the data filename

These get filled in for you when you enter the data dimensions – change them if you want

Enter the spatial dimensions

Doesn’t matter here, but a good value for other data

Describe the data at each point (type and data dimension)
This saves the .general file, which will eventually tell OpenDX where to find the data and how to handle it.
The OpenDX .general File for a 3D Scalar Dataset

file = Z:\CS419h\pts.dat
grid = 32 x 32 x 32
format = ascii
interleaving = field
majority = row
field = field0
structure = scalar
type = float
dependency = positions
positions = regular, regular, regular, 0, 1, 0, 1, 0, 1
end

Regular positions

Scalar

3D
Grid or Scattered file (General Array Format)

- Grid type: [ ]
- Number of variables: [ ]
- Positions in data file: [ ]
- Single time step: [ ]
- Data organization: [ ]
- Browse Data: [ ]
- Test Import: [ ]
- Visualize Data: [ ]
- Describe Data: [ ]

Spreadsheet format file

Hints

Message Window

Begin Execution
Begin Execution

ECHO:

Object Description:
Input object is a Field, the basic data carrying structure in DIII.
There are 32768 data items, each is of type float (4-byte or real*4).
The positions are enclosed within the box defined by the corner points:
[0 0 0] and [31 31 31]
Data range is:
minimum = 0.01, maximum = 339.48, average = 25.4054
Input is not ready to be rendered because the Field does not have colors yet.
Use the ‘AutoColor’, ‘AutoGreyScale’, or ‘Color’ modules to add colors.
See next page
This all gets created automatically! Pretty amazing, huh?
Terrain Visualization .general File

file = Z:\OpenDX\or.dat
grid = 201 x 105 2D
format = ascii
interleaving = record
majority = column
field = field0
structure = scalar Scalar
type = float
dependency = positions
positions = regular, regular, 0, 1, 0, -1
end

Regular positions
Vector Field Visualization .general File

```
file = Z:\OpenDX\vecs.dat
grid = 16 x 16 x 16 3D
format = ascii
interleaving = record-vector
majority = column
field = velocity
structure = 3-vector 3-element Vector
type = float
dependency = positions
positions = regular, regular, regular, 0, 1, 0, 1, 0, 1
end
```

Regular positions
Visualizing Points on a Scattered Grid (e.g., Digital Elevation Mapping)

<table>
<thead>
<tr>
<th>West→East</th>
<th>South→North</th>
<th>Elevation</th>
</tr>
</thead>
<tbody>
<tr>
<td>-360.78</td>
<td>128.438</td>
<td>2.25</td>
</tr>
<tr>
<td>-360.75</td>
<td>128.428</td>
<td>2.31</td>
</tr>
<tr>
<td>-360.80</td>
<td>128.405</td>
<td>2.20</td>
</tr>
<tr>
<td>-360.81</td>
<td>128.370</td>
<td>1.99</td>
</tr>
<tr>
<td>-360.91</td>
<td>128.369</td>
<td>1.75</td>
</tr>
<tr>
<td>-361.00</td>
<td>128.359</td>
<td>1.65</td>
</tr>
<tr>
<td>-361.16</td>
<td>128.354</td>
<td>1.77</td>
</tr>
<tr>
<td>-361.21</td>
<td>128.344</td>
<td>1.70</td>
</tr>
<tr>
<td>-361.25</td>
<td>128.344</td>
<td>1.76</td>
</tr>
</tbody>
</table>
Scattered Data (irregular positions, irregular connections)

Data Dimension: a scalar value at each position

Spatial Dimension: 2D

See next page
File → Save as...
A Scattered Grid .general File

file = Z:\OpenDX\exportxyz.dat
points = 1022  A 1D list
format = ascii
interleaving = field
field = locations, field0  Each entry has a location and data values
structure = 2-vector, scalar
type = float, float
end

Each location has 2 values (X,Y). Each data is a single value (scalar).
ERROR: Import: Bad parameter can't open file 'Z:\CS419\pts.dat'

Begin Execution:
Object Description:
Input object is a Field, the basic data carrying structure in DX.
There are 1022 data items, each is of type float (4-byte or real*4).
The positions are enclosed within the box defined by the corner points:
[-368.84 -142.418 0] and [408.12 131.769 0]
Data range is:
minimum = 0, maximum = 7.72, average = 3.01657
Input is not ready to be rendered because the Field does not have colors yet.
Use the ‘AutoColor’, ‘AutoGreyScale’, or ‘Color’ modules to add colors.

ECHO:
Visualizing Points from a Scattered Grid (e.g., Digital Elevation Mapping)

Connect creates triangles from scattered points.

ShowConnections creates graphics to show how the scattered point connections were made.

Sample Terrain Surface
Regridding a Dataset
(especially good for writing out a Connect’ed Scattered Grid as a Rectangular Array)

You might be doing this to downsize a dataset or to create a regular grid from a scattered grid

If Scattered data, route through Connect first.

Creates a new grid to project the data onto

Projects the original data onto the new grid

Writes out the new grid data file
Regridding a Dataset

Coordinates at the lower corner of the dataset
Step size in each dimension
How many steps to make in the grid

Note: the upper corner coordinates will be “origin + (counts-1)*delta”
Writing a .general Dataset as a Native OpenDX .dx Dataset

The .dx file format embeds the data description information, the positions, and the data values in one file. This makes it easier to keep track of and easier to give to other people.

Says that you want to Export the data in a native OpenDX .dx file form.