Data Visualization
NIH-NSF BBSI: Simulation and Computer Visualization of Biological Systems at Multiple Scales
June 2-4, 2004
Joel R. Stiles, MD, PhD

What is the goal?
A generalized environment for manipulation and visualization of multidimensional data

More generally -
A means to map N-dimensional data onto 2-D or 3-D spaces, and visualize as a 2-D projection

Examples of Common Datasets:
- Atmospheric data
- Oceanographic data
- Geological data
- Genomic sequences
- Protein sequences
- Protein structures
- Light & electron microscope images
- Medical imaging (CAT, MRI, PET, Ultrasound, etc.)
- Models
- Simulation data

Day 1
1. Introduction to Computer Animation
2. Introduction to OpenDX

Day 2
3. Overview of Computer Rendering
4. Multidimensional Data Representation and Manipulation

Day 3
5. Overview of Common Techniques for Multidimensional Data Visualization
6. Overview of Stereographic Visualization
1. Introduction to Computer Animation

(Examples: Issues and Answers)

2. Introduction to OpenDX (www.opendx.org)

- A “Complete Visualization Environment”
- Conceptually based on underlying abstract data model
- Three visual programming support components:
  - Graphical program editor - visual programs
  - Core set of supplied data transformations – modules
  - Client-server execution model – user interface separate from rendering engine (DX executive)
- Advanced features:
  - User-defined macros
  - Scripting language
  - Full API (Application Programming Interface)
How does a computer make a picture?
The computer renderer works like a virtual pinhole camera.

\[ \tan(\theta) = \frac{h_o}{o} = \frac{h_i}{f}, \text{ so } h_i = \frac{f}{o} h_o \]
4. Multidimensional Data Representation and Manipulation

- Atmospheric data
- Oceanographic data
- Geological data
- Genomic sequences
- Protein sequences
- Protein structures
- Light & electron microscopy images
- Medical imaging (CAT, MRI, PET, Ultrasound, etc.)
- Models
- Simulation data

Measurement, Modeling, Simulation, Visualization Project Flow

- Primitive Surfaces:
  - Quadric - sphere, cone, cylinder, hyperboloid, paraboloid, torus
  - Parametric - bilinear & bicubic patches, patch surfaces, non-uniform rational B-spline surfaces (NURBs)

- Discretized Meshes:
  - Linear - line element primitives
  - Surface - quad, triangle primitives
  - Volume - tetrahedron, cuboid, hexahedron, prism primitives
Discretized Meshes
Discretized Meshes (Grids) are characterized by their dimensionality and the pattern of connections between points:

- **Regular** – defined by an origin point, deltas (distance between points) in each dimension, and counts (number of points) in each dimension

- **Deformed Regular** – regular connections between points that do not have constant linear deltas

- **Irregular** – set of points and explicitly defined connections

Data Dependency
Data values (integer, real, complex, scalar, vector, matrix, tensor, text,...) can be mapped to either grid points (position-dependent) or grid elements (connection-dependent).

Position-dependent = node-centered, location-centered
Data Dependency

Data values (integer, real, complex, scalar, vector, matrix, tensor, text, ...) can be mapped to either grid points (position-dependent) or grid elements (connection-dependent).

Connection-dependent = cell-centered

OpenDX Data Model

An N-dimensional abstract data space from which the user takes 2-D and 3-D visual “snapshots” to create viewable images.

Uses an object-oriented, self-describing approach to defining the datasets imported, used, and manipulated by the system.

OpenDX Data Model

Generally uses 6 types of descriptive objects:

1. Attribute: names an association between an OpenDX object (array, component, field, or group) and a (simple or compound) value. A typical use for an attribute is to associate “metadata” with a data set.

2. Array: a basic data carrying structure that holds actual data. OpenDX uses one-dimensional arrays and permits the array elements to be of any type, so an array object can be described simply by listing the number of items it contains. Array elements are referenced by index.

3. Component: an element of a field with a specific role in data description; a component is typically associated with an array object with a specific associated name.

4. Field: a fundamental compound object in OpenDX, used to collect and encapsulate related components. All its elements must be components.
OpenDX Data Model

Generally uses 6 types of descriptive objects:

5. **Group**: compound object used to collect *members* that themselves may be fields and/or groups; it cannot collect components (a field is used for that purpose). A member of a group may be referenced either by name or index.

6. **Special**: used to describe special attributes or characteristics of objects used in the rendering process, e.g., Camera, Light, Transform, etc.

**OpenDX Data Model**

**Attributes**:

Formalize the attachment of metadata to specific parts of a data set. Examples of predefined attributes:

- "data" specifies the component on which the given component depends, e.g., a "data" component can be dependent upon "positions".
- "ref" specifies the component to which the given component refers, e.g., a "connections" component will typically refer to the "positions" component.
- "der" specifies that a component is derived from another component, and so should be recalculated or deleted when the component it is derived from changes, e.g., the "box" component typically has a "der" attribute naming the "positions" component.
- "element type" is an attribute of the "connections" component, and names the type of interpolation primitive.
- "shade" indicates whether or not to shade the object if a "normals" component is present.

**OpenDX Data Model**

**Array Objects**:

- Items are referenced consecutively starting at zero.
- "type" attribute describes the internal numerical format to be used for the array’s data. Predefined type values include double, float, int, uint, short, ushort, byte, ubyte, and string.
- "category" attribute specifies which of two possible floating point representations is to be used, real or complex.
- "rank" attribute refers to element order dimensionality, where rank 0 indicates a scalar, 1 a vector, 2 a matrix or rank-2 tensor, and 3 or higher a higher-order tensor.
- "shape" attribute defines the dimensionality in each of the order dimensions of the structure. Thus, for rank-0 items (scalars), there is no shape. For rank-1 structures (vectors), the shape is a single number corresponding to the number of dimensions. For rank-2 structures, shape is two numbers, and so on.
OpenDX Data Model

Field Objects consist of component arrays. Typical predefined field components:

- “positions” stores the coordinates of a set of positions in an n-dimensional space.
- “connections” provides a means for explicitly relating individual collections of positions (e.g., representing lines, surfaces, etc.) and interpolating data values between positions.
- “data” stores actual data values. Only one component can be named “data” in a field, but other components can be used to store alternate data and can be switched with existing “data” at any time.
- “box”, “colors”, “front colors”, “back colors”, “normals”, “opacity”, “opacities”, etc., provide specific information that directs the renderer’s operation.

OpenDX Data Model

Group Objects consist of members. There are four specific group types:

- “Generic” group (standard).
- “Multigrid” group is a collection of separate fields, each with its own grid (with common element type) but treated as a single field, rather than as a group.
- “Composite field” group is similar to multigrid group, used primarily to segment fields to permit parts of the field/group to be processed in parallel.
- “Series” group is a generic group that stores a series value (e.g., time step) for each member.