Data Visualization

NIH-NSF BBSI: Simulation and Computer Visualization of Biological Systems at Multiple Scales

Joel R. Stiles, MD, PhD

PITTSBURGH SUPERCOMPUTING CENTER

NRBSC

What is real?

Examples of some mind-bending optical illusions enabled by computer graphics.

What is the goal?

A generalized environment for manipulation and visualization of multidimensional data

More generally -

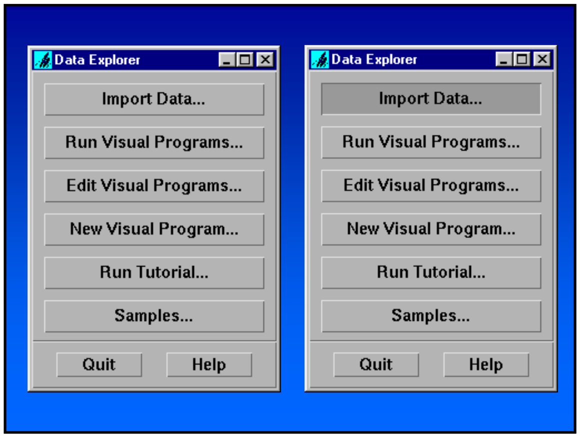
A means to map N-dimensional data onto 2-D or 3-D spaces, and visualize as a 2-D projection

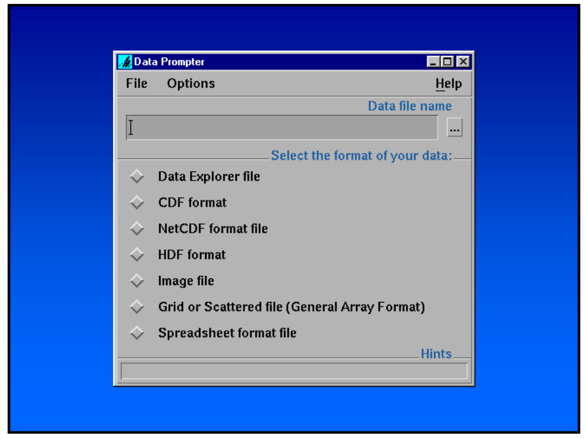
Examples of Common Datasets:

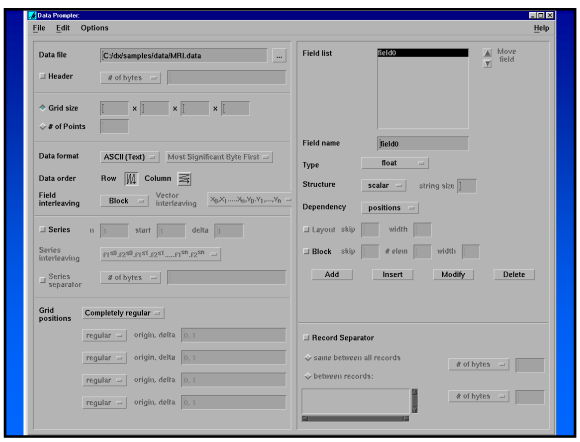
- Atmospheric data
- Oceanographic data
- Geological data
- Genomic sequences
- Protein sequences
- Protein structures
- Light & electron microscope images
- Medical imaging (CAT, MRI, PET, Ultrasound, etc.)
- Models
- Simulation data

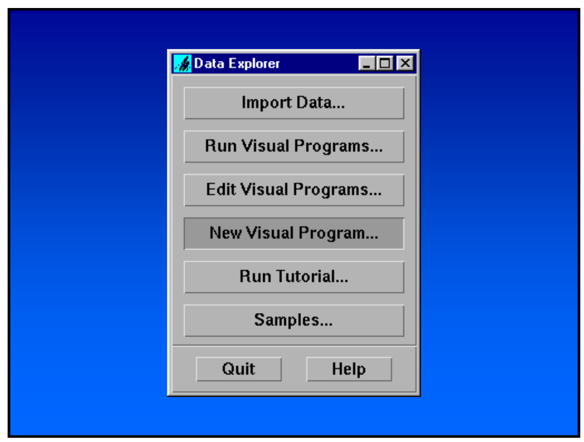
Introduction to OpenDX (www.opendx.org)

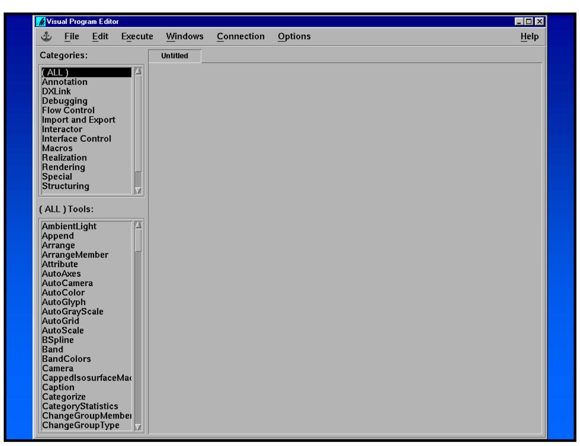
- A "Complete Visualization Environment"
- Conceptually based on underlying abstract data model
- Three visual programming support components:
  - Graphical program editor - visual programs
  - Core set of supplied data transformations – modules
  - Client-server execution model – user interface separate from rendering engine (DX executive)
- Advanced features:
  - User-defined macros
  - Scripting language
  - Full API (Application Programming Interface)

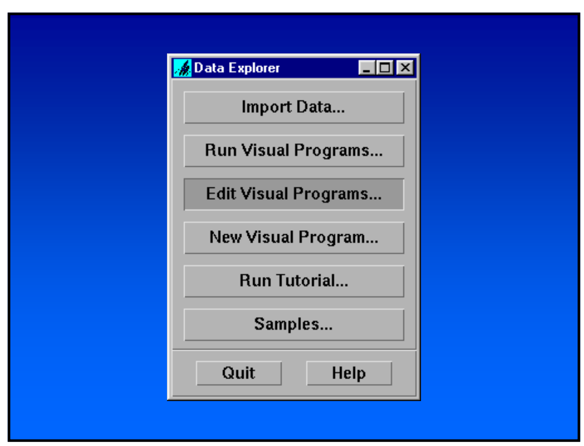

Data Explorer
Import Data...
Run Visual Programs...
Edit Visual Programs...
New Visual Program...
Run Tutorial...
Samples...
Quit
Help
Data Explorer
Import Data...
Run Visual Programs...
Edit Visual Programs...
New Visual Program...
Run Tutorial...
Samples...
Quit
Help

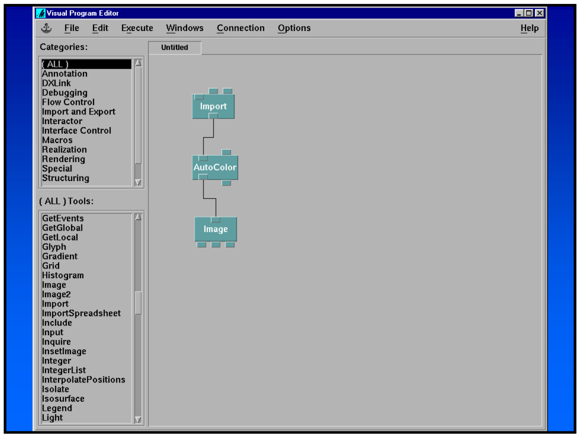

Data Prompter
File
Options
Help
Data file name
Select the format of your data:
Data Explorer file
CDF format
NetCDF format file
HDF format
Image file
Grid or Scattered file (General Array Format)
Spreadsheet format file
Hints

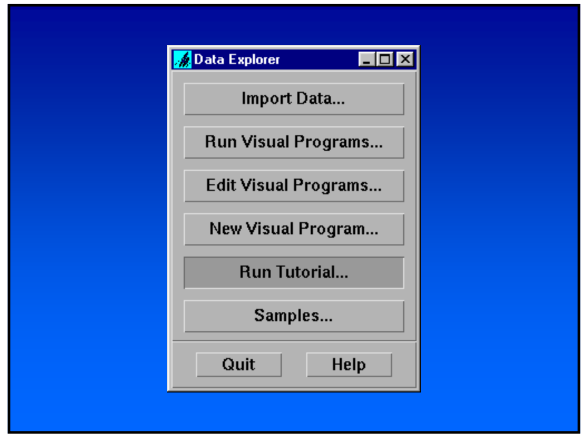

File
Edit
Options
Help
Data file
C:/dx/samples/data/MRI.data
Header
Grid size
# of Points
Data format
ASCII (Text)
Data order
Row
Column
Field interleaving
Block
Series
Grid positions
Completely regular
Field list
Field name
Type
float
Structure
scalar
Dependency
positions
Add
Insert
Modify
Delete
Record Separator

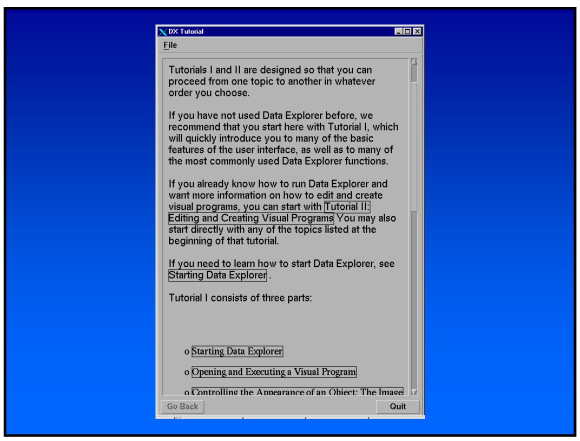

Data Explorer
Import Data...
Run Visual Programs...
Edit Visual Programs...
New Visual Program...
Run Tutorial...
Samples...
Quit
Help

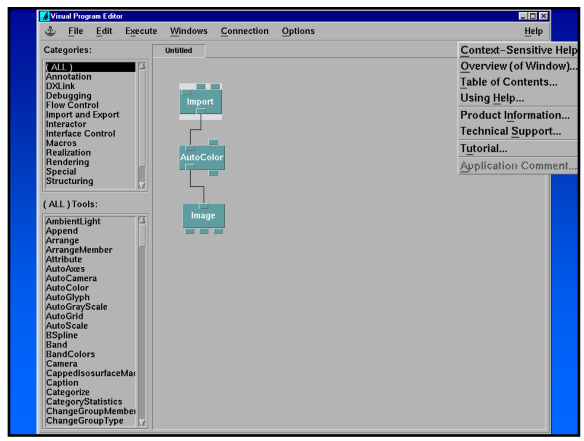

Visual Program Editor
File
Edit
Execute
Windows
Connection
Options
Help
Categories:
Untitled
(ALL)
Annotation
DXLink
Debugging
Flow Control
Import and Export
Interactor
Interface Control
Macros
Realization
Rendering
Special
Structuring
(ALL) Tools:
AmbientLight
Append
Arrange
ArrangeMember
Attribute
AutoAxes
AutoCamera
AutoColor
AutoGlyph
AutoGrayScale
AutoGrid
AutoScale
BSpline
Band
BandColors
Camera
Caption
Categorize
CategoryStatistics
ChangeGroupType

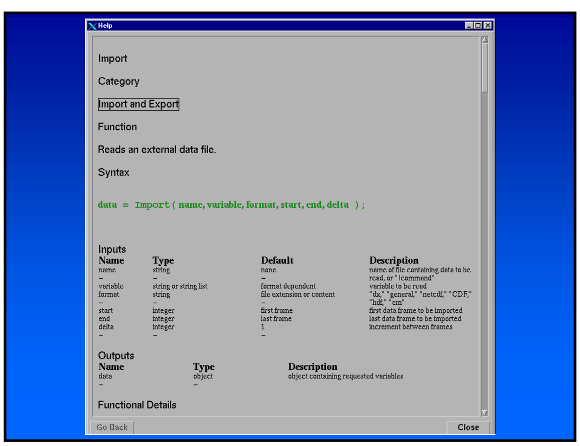

Data Explorer
Import Data...
Run Visual Programs...
Edit Visual Programs...
New Visual Program...
Run Tutorial...
Samples...
Quit
Help

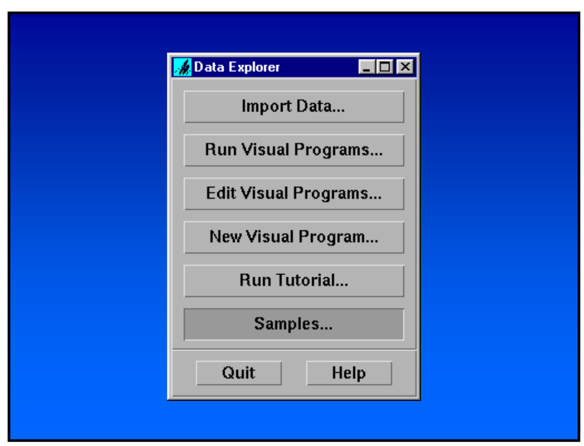

Visual Program Editor
File Edit Execute Windows Connection Options Help
Categories:
Untitled
(ALL)
Annotation
DXLink
Debugging
Flow Control
Import and Export
Interactor
Interface Control
Macros
Realization
Rendering
Special
Structuring
(ALL) Tools:
GetEvents
GetGlobal
GetLocal
Glyph
Gradient
Grid
Histogram
Image
Image2
Import
ImportSpreadsheet
Include
Input
Inquire
InsetImage
Integer
IntegerList
InterpolatePositions
Isolate
Isosurface
Legend
Light
Import
AutoColor
Image

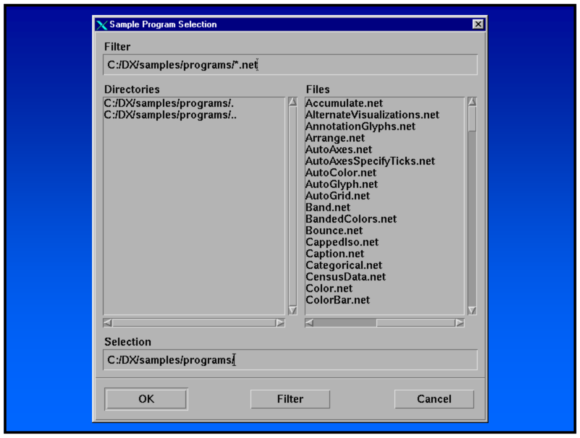

Data Explorer
Import Data...
Run Visual Programs...
Edit Visual Programs...
New Visual Program...
Run Tutorial...
Samples...
Quit
Help


DX Tutorial
File
Tutorials I and II are designed so that you can proceed from one topic to another in whatever order you choose.
If you have not used Data Explorer before, we recommend that you start here with Tutorial I, which will quickly introduce you to many of the basic features of the user interface, as well as to many of the most commonly used Data Explorer functions.
If you already know how to run Data Explorer and want more information on how to edit and create visual programs, you can start with Tutorial II: Editing and Creating Visual Programs You may also start directly with any of the topics listed at the beginning of that tutorial.
If you need to learn how to start Data Explorer, see Starting Data Explorer .
Tutorial I consists of three parts:
o Starting Data Explorer
o Opening and Executing a Visual Program
Go Back
Quit


Visual Program Editor
File Edit Execute Windows Connection Options Help
Categories:
Untitled
(ALL)
Annotation
DXLink
Debugging
Flow Control
Import and Export
Interactor
Interface Control
Macros
Realization
Rendering
Special
Structuring
(ALL) Tools:
AmbientLight
Append
Arrange
ArrangeMember
Attribute
AutoAxes
AutoCamera
AutoColor
AutoGlyph
AutoGrayScale
AutoGrid
AutoScale
BSpline
Band
BandColors
Camera
Caption
Categorize
CategoryStatistics
ChangeGroupType
Import
AutoColor
Image
Context–Sensitive Help
Overview (of Window)...
Table of Contents...
Using Help...
Product Information...
Technical Support...
Tutorial...
Application Comment...


Help
Import
Category
Import and Export
Function
Reads an external data file.
Syntax
data = Import (name, variable, format, start, end, delta) ;
Inputs
Name Type Default Description
name string none name of file containing data to be read, or "!command"
variable string or string list format dependent variable to be read
format string file extension or content "dx," "general," "netcdf," "CDF," "hdf," "cm"
start integer first frame first data frame to be imported
end integer last frame last data frame to be imported
delta integer 1 increment between frames
Outputs
Name Type Description
data object object containing requested variables
Functional Details
Go Back
Close

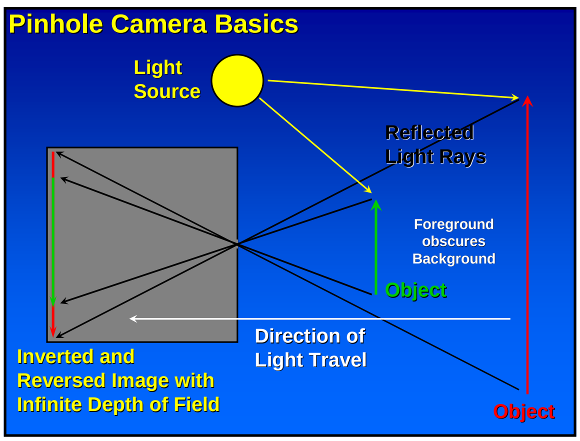

Data Explorer
Import Data...
Run Visual Programs...
Edit Visual Programs...
New Visual Program...
Run Tutorial...
Samples...
Quit
Help

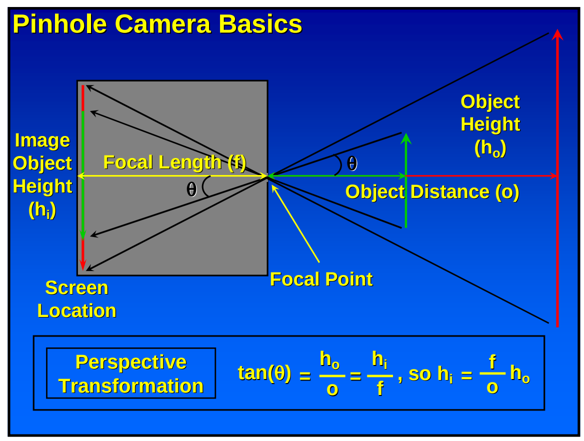


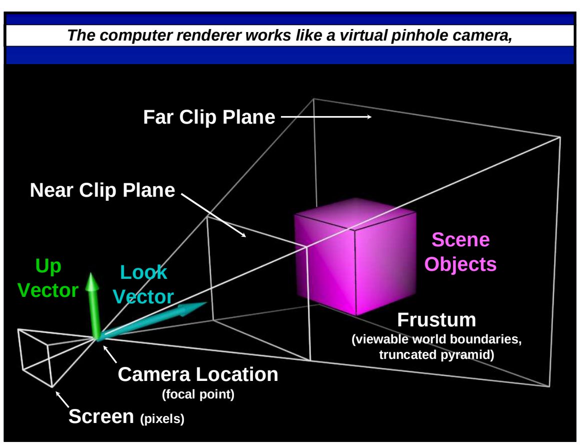
Overview of Computer Rendering

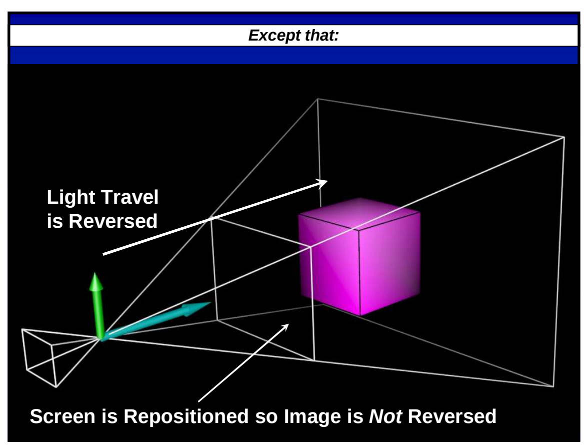
- or -

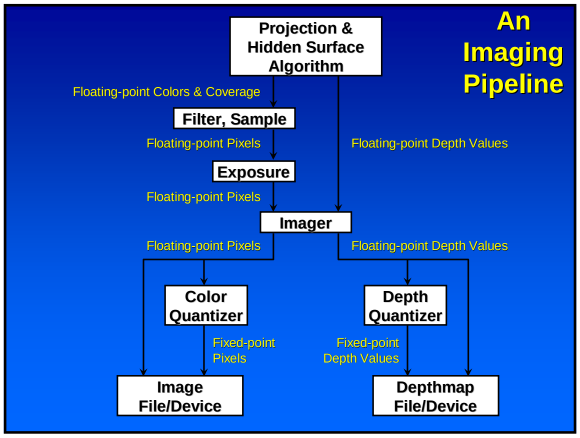
How does a computer make a picture?

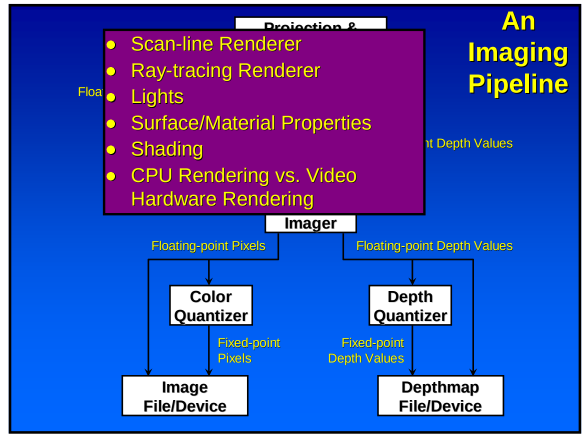


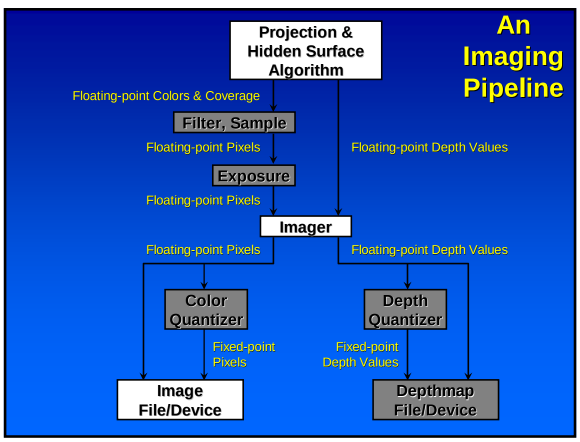


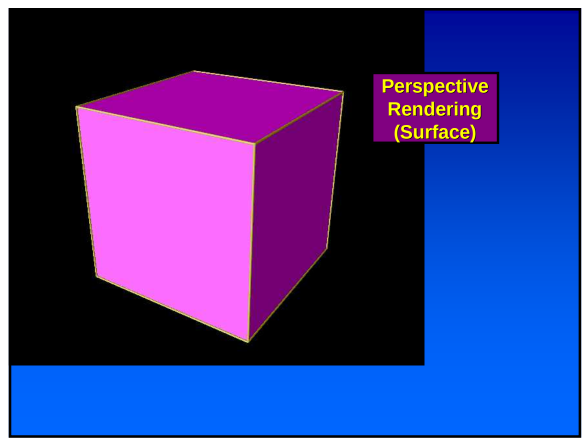


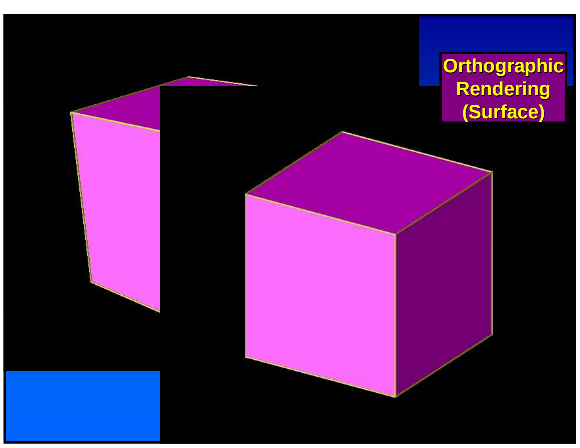


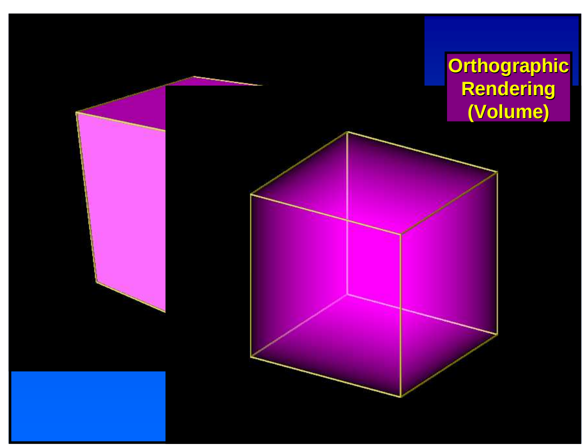

An Imaging Pipeline
Projection & Hidden Surface Algorithm
Floating-point Colors & Coverage
Filter, Sample
Floating-point Pixels
Floating-point Depth Values
Exposure
Floating-point Pixels
Imager
Floating-point Pixels
Floating-point Depth Values
Color Quantizer
Depth Quantizer
Fixed-point Pixels
Fixed-point Depth Values
Image File/Device
Depthmap File/Device

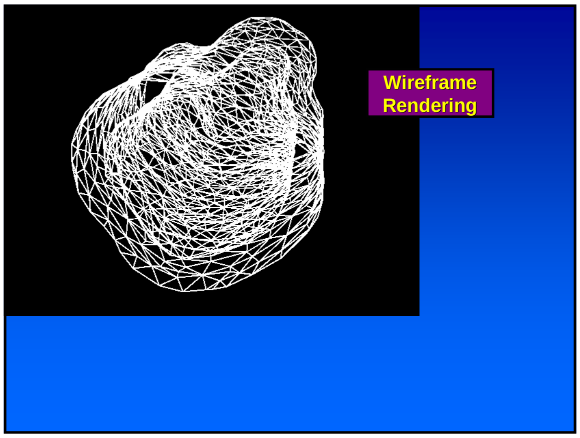

An Imaging Pipeline
Scan-line Renderer
Ray-tracing Renderer
Lights
Surface/Material Properties
Shading
CPU Rendering vs. Video Hardware Rendering
Imager
Floating-point Pixels
Floating-point Depth Values
Color Quantizer
Depth Quantizer
Fixed-point Pixels
Fixed-point Depth Values
Image File/Device
Depthmap File/Device

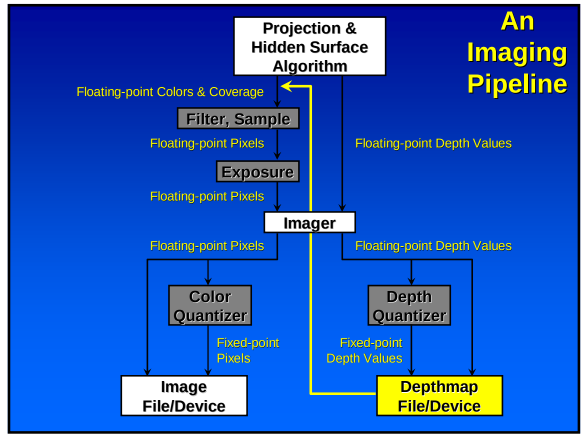

An Imaging Pipeline
Projection & Hidden Surface Algorithm
Floating-point Colors & Coverage
Filter, Sample
Floating-point Pixels
Floating-point Depth Values
Exposure
Floating-point Pixels
Imager
Floating-point Pixels
Floating-point Depth Values
Color Quantizer
Depth Quantizer
Fixed-point Pixels
Fixed-point Depth Values
Image File/Device
Depthmap File/Device

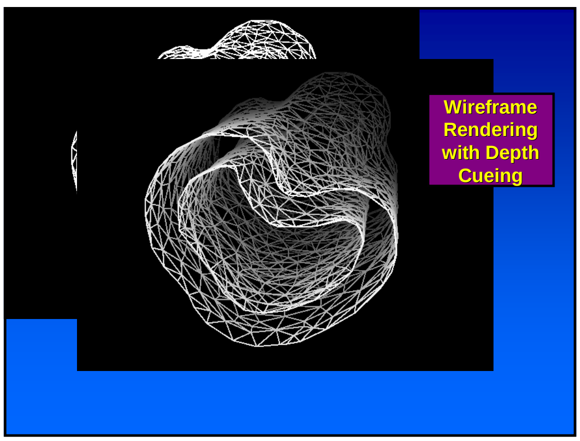

Perspective Rendering (Surface)

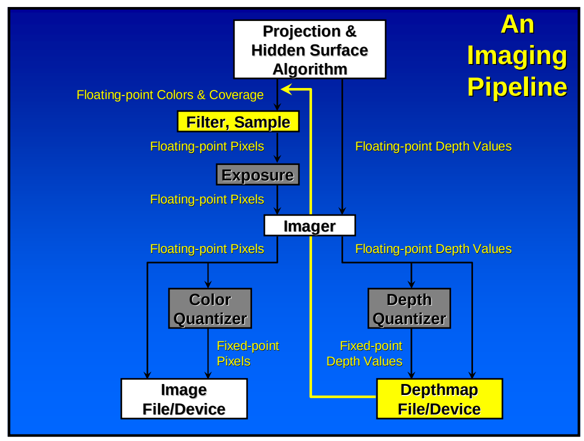

Orthographic Rendering (Surface)

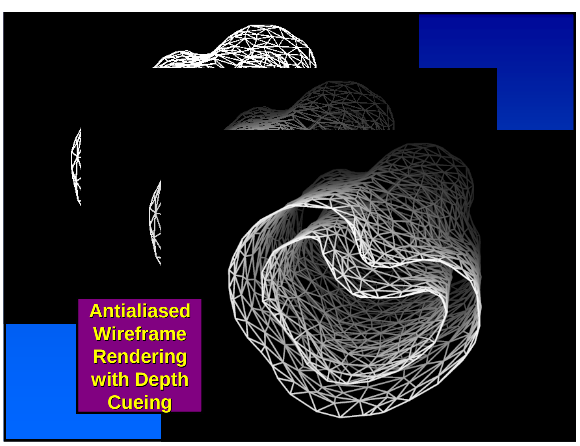

Orthographic Rendering (Volume)

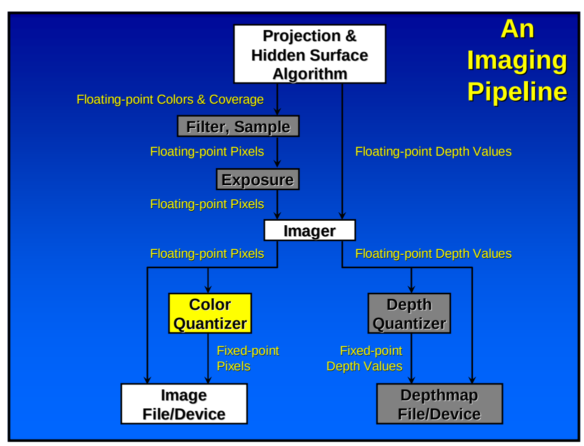

Wireframe Rendering

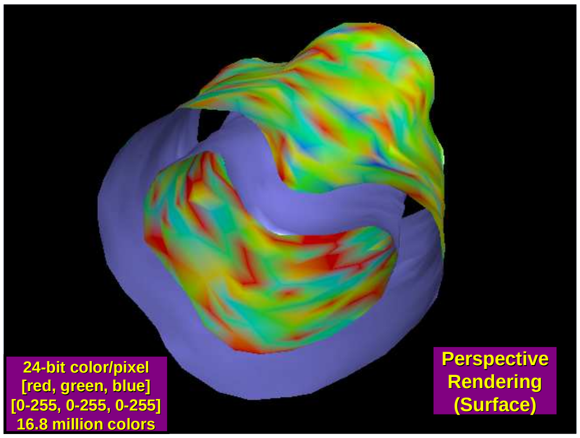

An Imaging Pipeline
Projection & Hidden Surface Algorithm
Floating-point Colors & Coverage
Filter, Sample
Floating-point Pixels
Exposure
Floating-point Pixels
Floating-point Depth Values
Imager
Floating-point Pixels
Floating-point Depth Values
Color Quantizer
Depth Quantizer
Fixed-point Pixels
Fixed-point Depth Values
Image File/Device
Depthmap File/Device

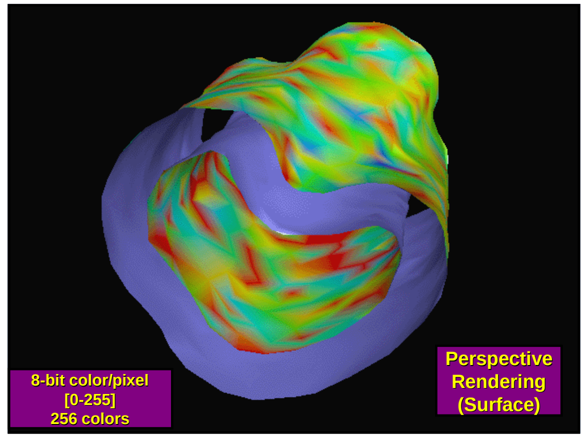

Wireframe Rendering with Depth Cueing


An Imaging Pipeline
Projection & Hidden Surface Algorithm
Floating-point Colors & Coverage
Filter, Sample
Floating-point Pixels
Exposure
Floating-point Pixels
Floating-point Depth Values
Imager
Floating-point Pixels
Floating-point Depth Values
Color Quantizer
Depth Quantizer
Fixed-point Pixels
Fixed-point Depth Values
Image File/Device
Depthmap File/Device


Antialiased Wireframe Rendering with Depth Cueing

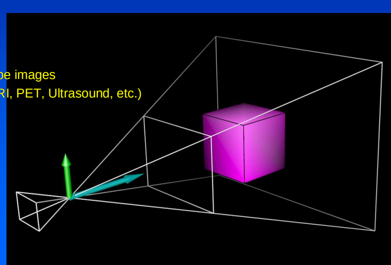

An Imaging Pipeline
Projection & Hidden Surface Algorithm
Floating-point Colors & Coverage
Filter, Sample
Floating-point Pixels
Exposure
Floating-point Pixels
Floating-point Depth Values
Imager
Floating-point Pixels
Floating-point Depth Values
Color Quantizer
Depth Quantizer
Fixed-point Pixels
Fixed-point Depth Values
Image File/Device
Depthmap File/Device



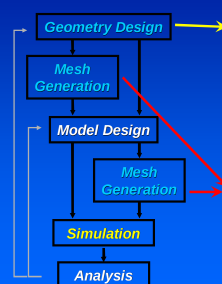


Multidimensional Data Representation and Manipulation

- 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



Measurement, Modeling, Simulation, Visualization Project Flow



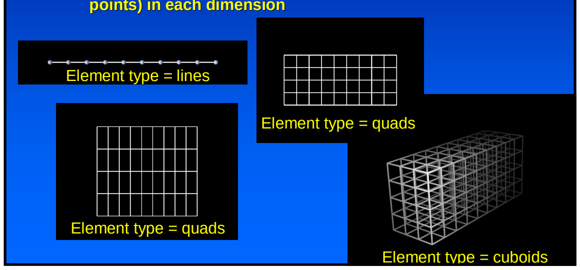
Primitive Surfaces:
Quadric - sphere, cone, cylinder, hyperboloid, paraboloid, torus
Parametric - bilinear & bicubic patches, patch surfaces, non-uniform rational B-spline surfaces (NURBs)

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:

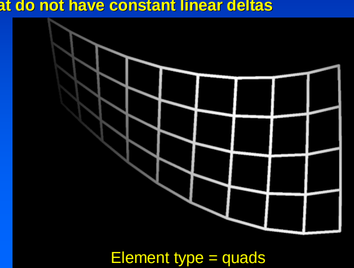
- **Regular** – defined by an origin point, deltas (distance between points) in each dimension, and counts (number of points) in each dimension



Discretized Meshes

Discretized Meshes (Grids) are characterized by their dimensionality and the pattern of connections between points:

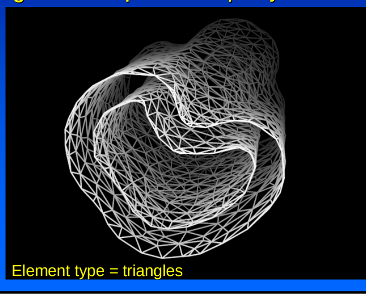
- **Deformed Regular** – regular connections between points that do not have constant linear deltas



Discretized Meshes

Discretized Meshes (Grids) are characterized by their dimensionality and the pattern of connections between points:

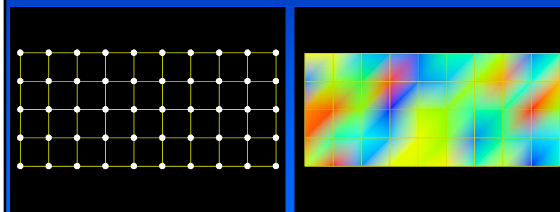
Irregular – set of points and explicitly defined connections



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).

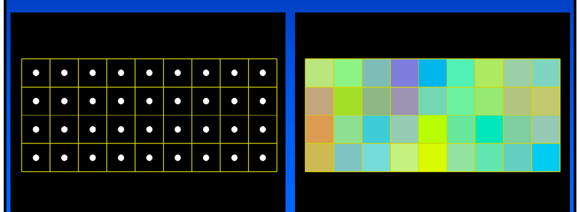
Position-dependent = node-centered, location-centered



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.

OpenDX Data Model

Generally uses 6 types of descriptive objects:

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

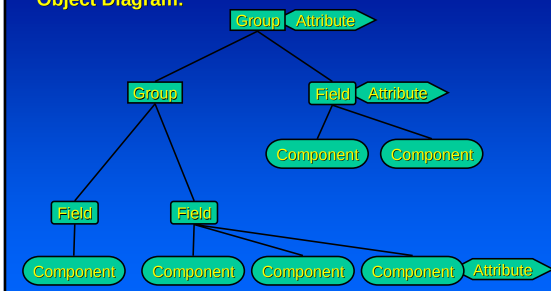
Generally uses 6 types of descriptive objects:

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

Object Diagram:



OpenDX Data Model

Attributes:

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

- "dep" 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.