MANCHESTER 1824

The University of Manchester

AVS/Express Visualization Edition

An Introductory Course

Tobias Schiebeck

Open to Europe Intensive Programme – TRABHCI

29th March - 9th April 2011

Universidad Politécnica de Valencia

What is AVS/Express?

- Object-oriented visualization development tool
- Modular
- Hierarchical
- Open and extensible
- Many predefined application components
  - *analyze*
  - *display*
  - *manipulate*
  - *interact*

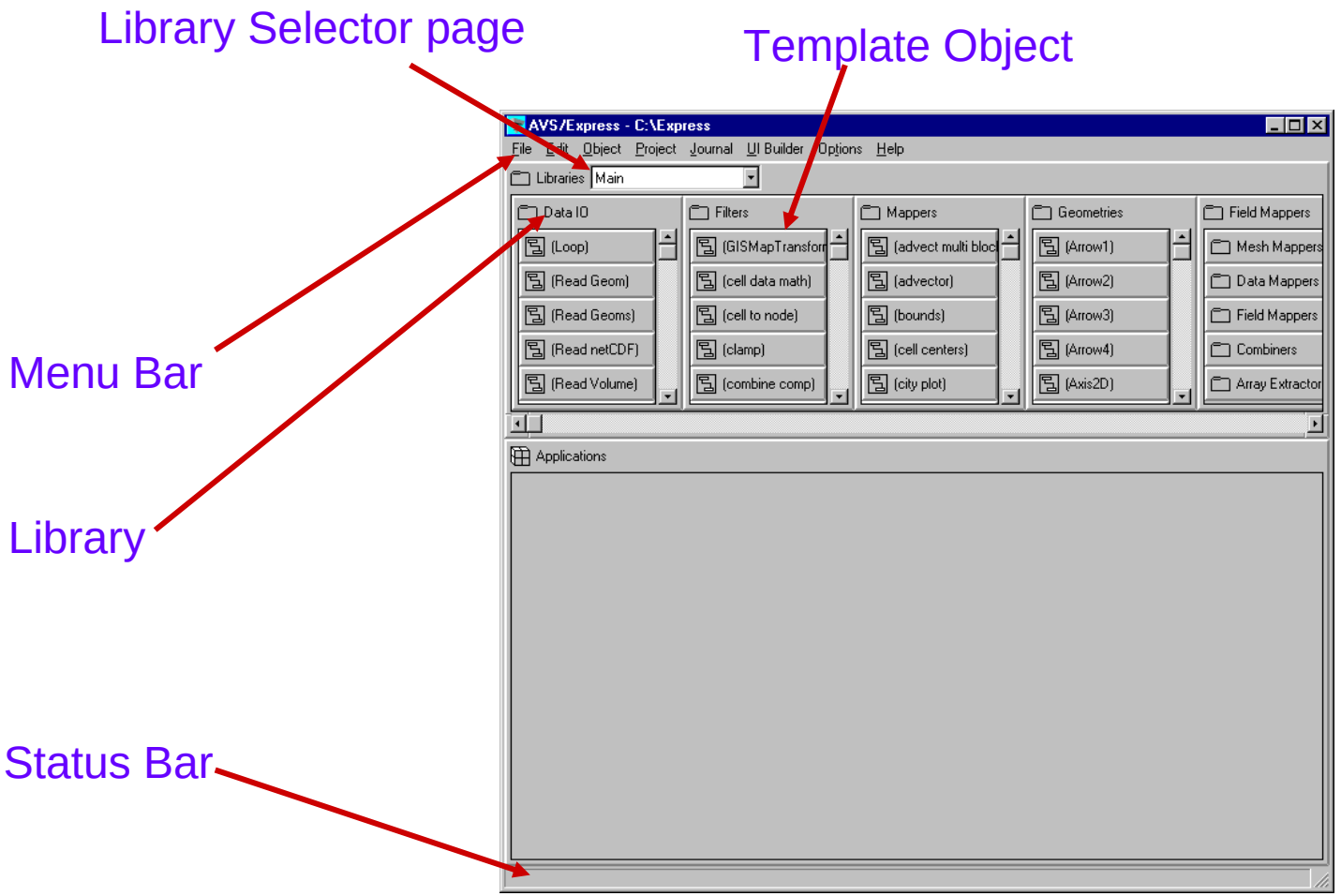
Developers and Visualization Edition

- There are three different editions of AVS/Express
  - *Visualization Edition - produce networks for a particular visualization*
  - *Developers Edition - producing custom applications within your organisation or for external customers*
  - *MultiPipe Edition – extension to the Developers edition for virtual environments*
- All have the same architecture
- Throughout the course we will use AVS/Express Visualization Edition

Working with AVS/Express

- Network Editor
- V Code
- Application Programmer Interface

The Network Editor



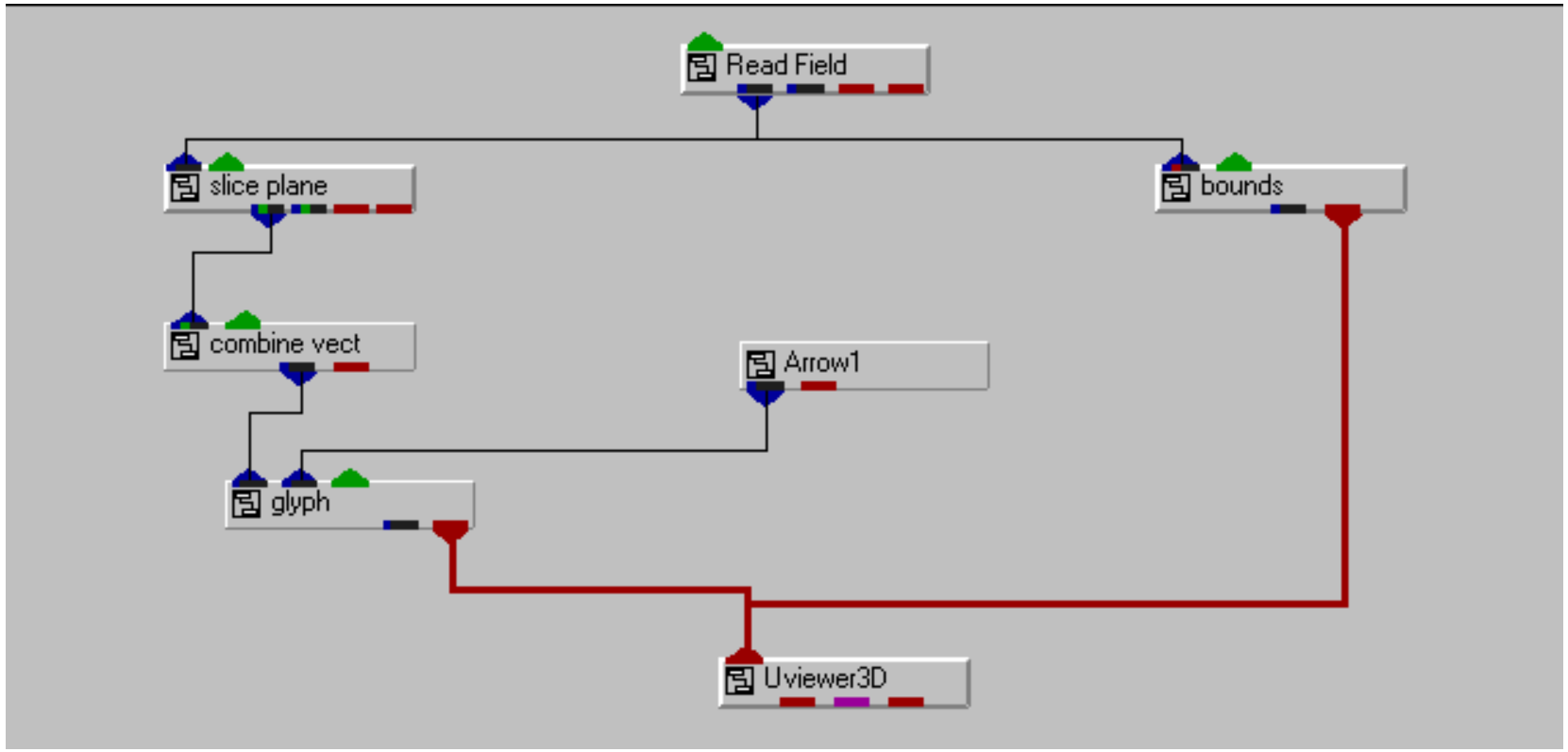
The Network Editor

- A visual programming environment for constructing entire range of objects
  - *simple groups*
  - *entire applications*
- Navigation
  - *Finding objects*
  - *Instantiation*
  - *Connecting*
  - *Disconnecting*
  - *Grouping*
  - *Scaling and positioning*

Visual Programming

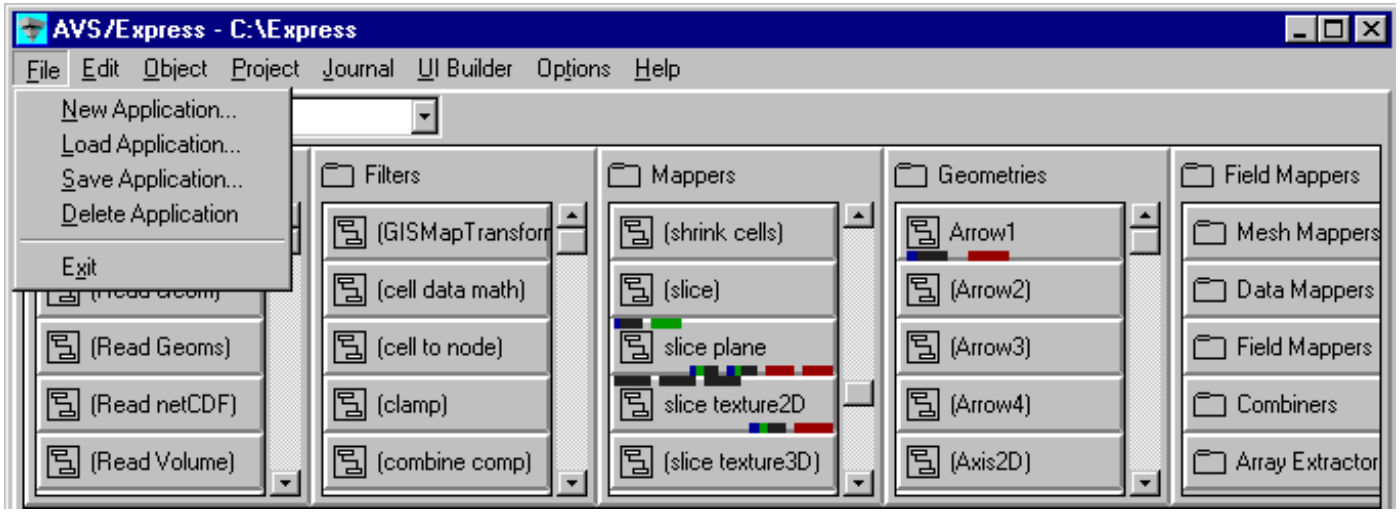
- Libraries of objects
- Application workspace
- Objects ‘dragged and dropped’ into the workspace
- Lines connecting objects represent data references
- Connections also drive execution of an application

An Example Application



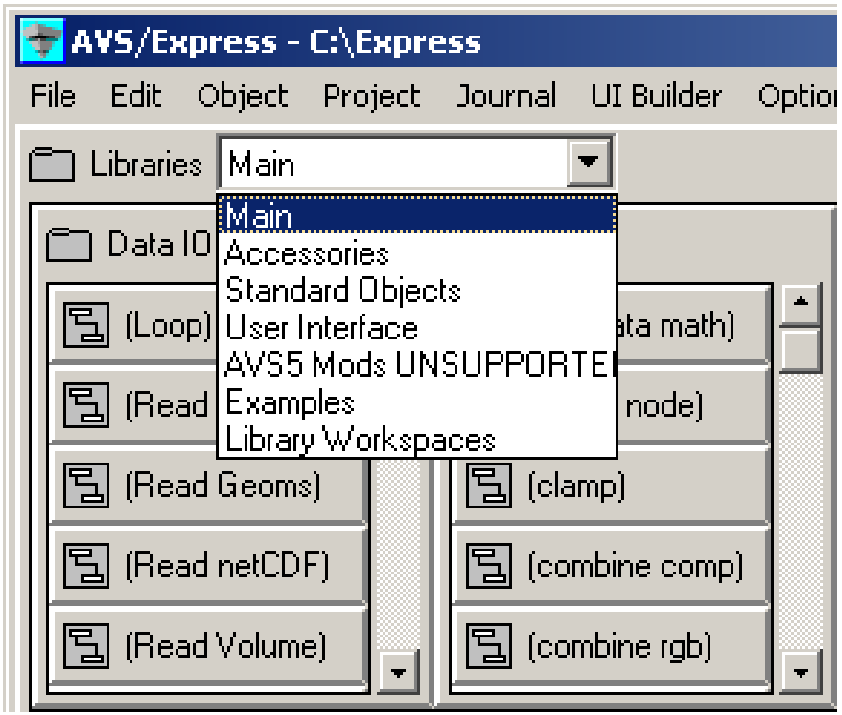
Network Editor Pulldown Menus

- File
- Edit
- Object
- Project
- Journal
- UI Builder
- Options
- Help



Libraries

- Main
- Accessories
- Standard Objects
- User Interface
- AVS5 Modules
- Examples
- Library Workspace



Data Viewer

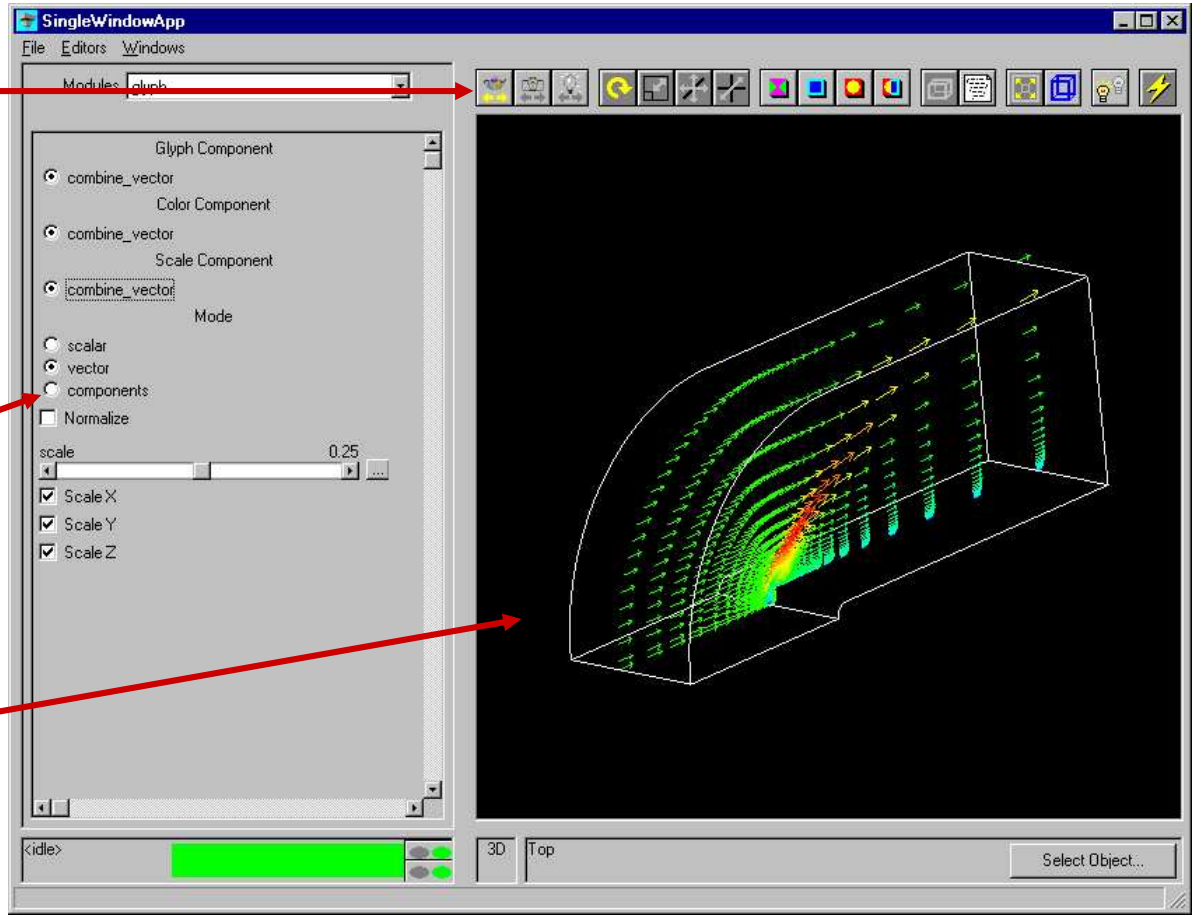
- Integrated viewing framework consisting of:
  - Data View Window for display
  - User Interface to manipulate objects
  - User interface to control viewer characteristics
- We will use 3D Applications (SingleWindowApp)

Data Viewer

Toolbar

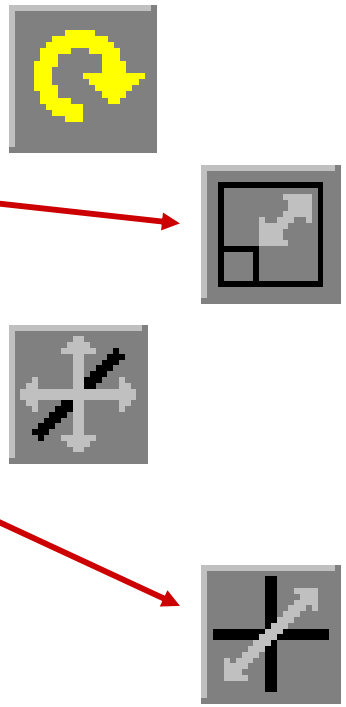
User Interface controls objects in Network Editor and window parameters

Window



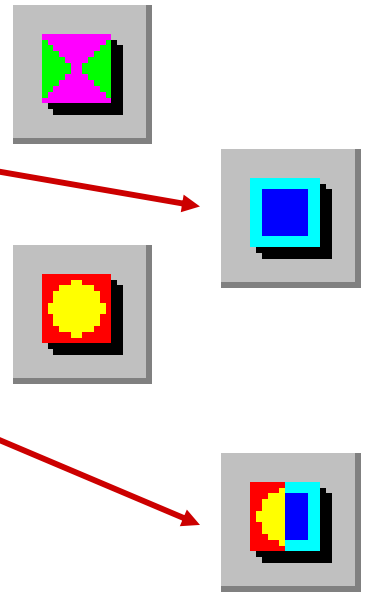
Using the Toolbar

- Transform Mode
  - *rotate*
  - *scale*
  - *XY translate*
  - *Z translate*
- Acts on
  - *Current object*
  - *Current light*
  - *Current camera*



Using the Toolbar II

- Reset
- Normalize
- Centre
- Perspective toggle
- Module Editor
- View Editor
- Camera Editor
- Object Editor



MANCHESTER 1824

The University of Manchester

Exercise I

Network Editor

MANCHESTER 1824

The University of Manchester

Importing Data & Visualization Techniques

Importing data into AVS/Express

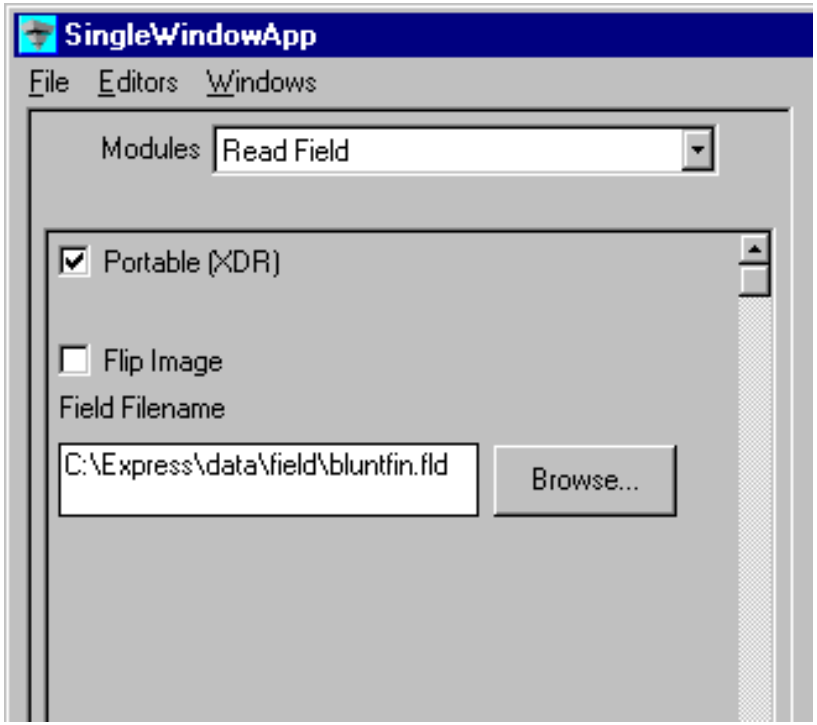
- Introduction
- The overall scheme
- Describing the data
- Some examples

Introduction

- When you import data in any visualization tool you need to know your data.
- Information you need:
  - Has the visualization tool a native reader for your data?
  - What format is your data:
    - *Computational space (1D, 2D, 3D, 4D …)*
    - *Coordinate space (2D, 3D)*
    - *Mapping between computational and coordinate space (uniform, rectilinear, irregular)*
- What do you expect to see?

Read_Field Module

- The Read_Field module has a file browser to select the header file



Overall Scheme

- The AVS field datatype is one of many ways to represent arrays of data in AVS/Express
- This data type can be read from an AVS field file using the Read_Field module
- The data is described by an ASCII header file
  - containing the data
  - reference the data via external files
- Referencing files is a better way of working

Describing the Data

- You must specify the dimensions of the data
  - *1D: list of scattered data*
  - *2D: Image*
  - *3D: volume of data*
- These specify computational space
  - *specified by keyword, ndim*
  - *Represents the shape of the data in the array*
  - *1D -> ndim = 1*
  - *2D -> ndim = 2 etc*

Describing the Data II

- You must specify dimensions as the data is passed to AVS/Express
- These specify the coordinate space
  - *specified by keyword, nspace*
  - *represents the physical dimensions; the number of coordinates required to represent a certain point*

Mapping Methods

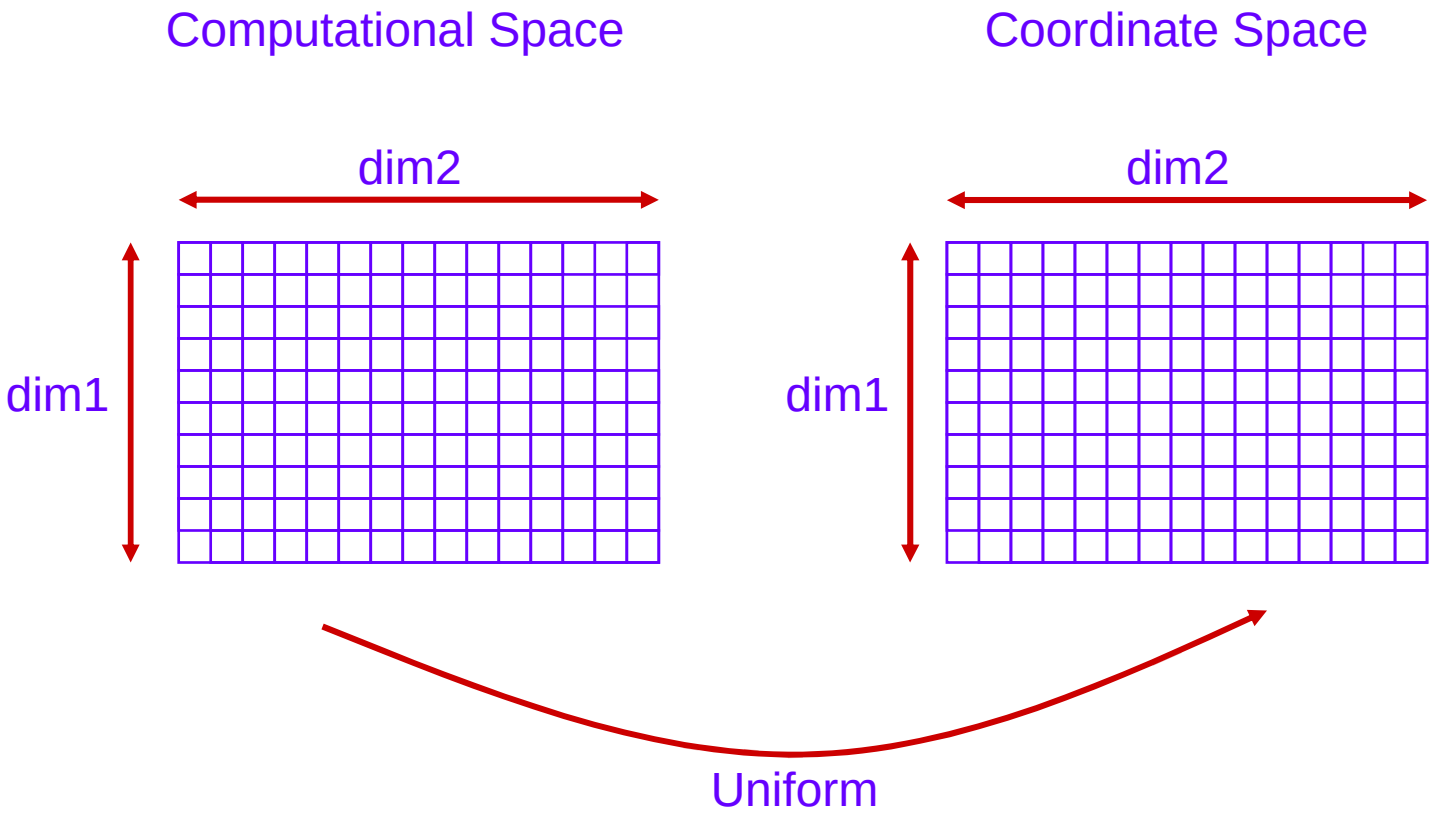
- Uniform
  - used to import regular grids of data
  - The coordinates are orthogonal and equidistant (equal spacing)
- Rectilinear
  - each data point is mapped to a physical coordinate
  - orthogonal Axes but not equidistant (unequal spacing)
- Irregular
  - dimensions of computational and coordinate space can be different
  - Each data value is mapped to a specified point in arbitrary space

Mapping Methods

Uniform

- The data is not transformed as it is imported (direct & implicit)
- The dimensions of the computational and coordinate space should be the same
- Used to import regular arrays of data
- No need to supply physical coordinates

Uniform Mapping

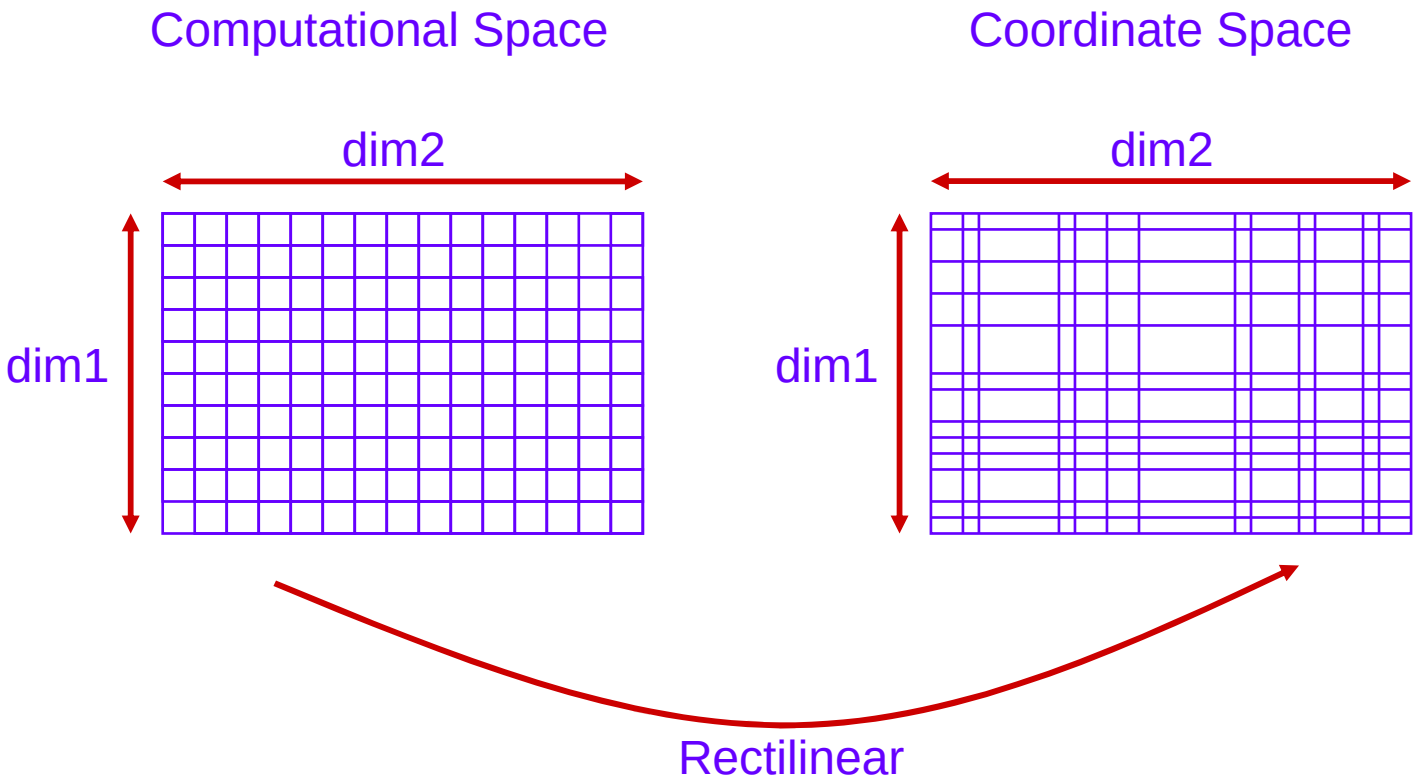


Mapping Methods

Rectilinear

- Each dimension in the data has an explicit coordinate mapping
- The space along the coordinate axes can be non-uniform
  - *e.g., logarithmic axes*
- The dimensions of the computational and coordinate space should be the same.

Rectilinear Mapping

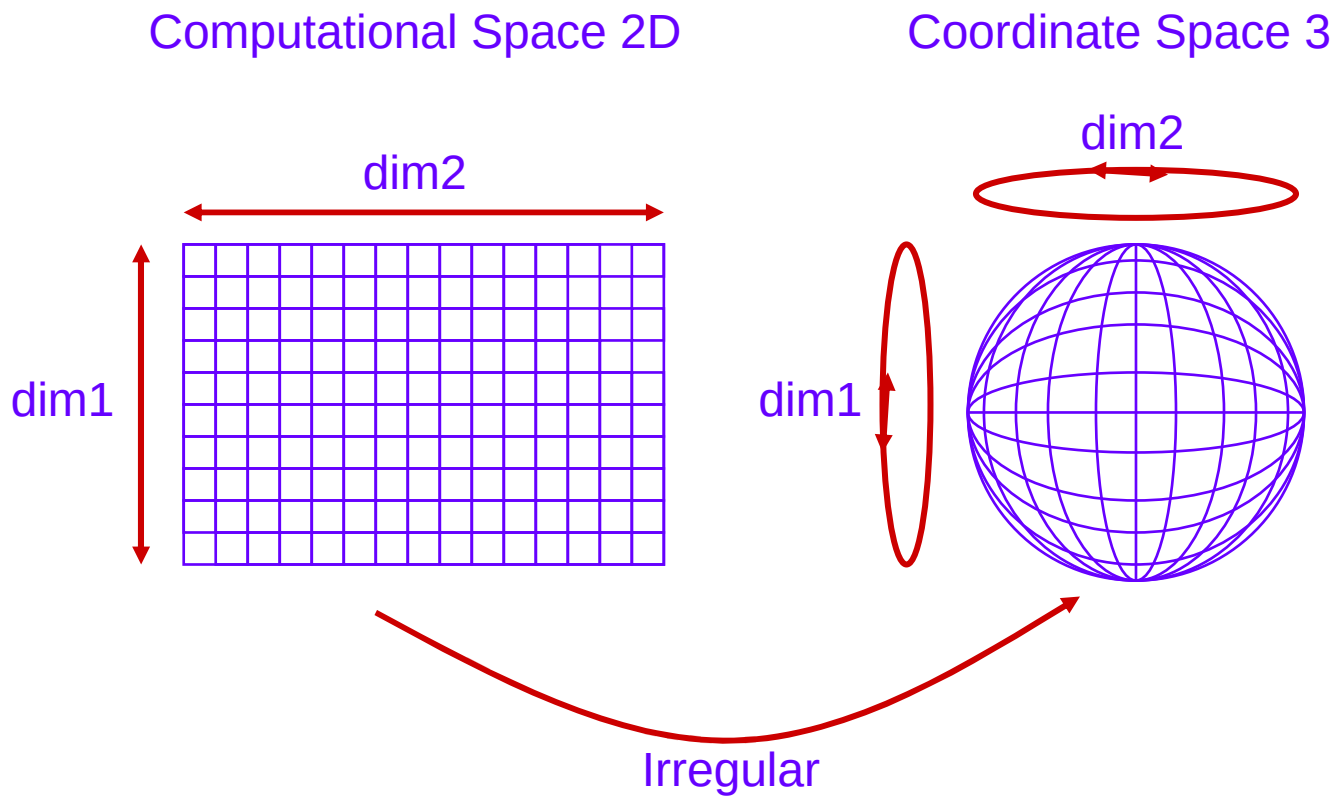


Mapping Methods

Irregular

- Each data element is mapped to a point in the coordinate space
  - *coordinates must be supplied*
- The dimensions of the computational and coordinate space can be different
  - *e.g., 1D scattered data in 3D space*
- Sometimes called Curvilinear Coordinates

Irregular Mapping



Data Element

- The number of data components
- The datatype
  - *byte, short, integer, float, double*
- Mapping method
- Location and layout of the data files
- A description of each data component
  - *specify the name*
  - *specify the units*

M1 data type - byte,integer,single or double

- the new version of avs/express - 6.4 can do 64 bit long - but machine has to be 64 bit
on an ordinary pc long will be 32 bit
McDerby, 25/04/2005

Header File

- The file should have an extension of .fld
- The first line has to be:

```
#AVS field file
```

- The rest of the file contains a number of keywords and values
- Comment lines begin with a #

Keywords

- ndim = number of dimensions in computational space
- dim1 = specify the size of dimension 1
  - *can be thought of as the x dimension*
- nspace = number of dimensions in coordinate space
- veclen = number of data components per element
- data = type of data component
- field = mapping method
- label = label for each data component
- unit = units for each data component

Reading the Data

- Once the data structure has been defined you specify the location and format of the data file(s).
  - *The data file(s) can be binary or ASCII*
- Each data component needs a line:
  - **`variable`** *N <number of format keywords>*
- Each item of coordinate information needs a line:
  - **`coord`** *N <number of format keywords>*
- Each line can reference a different file

Reading the Data II

- There are five format keywords for reading the data
  - *file = location of data file*
  - *filetype = format of the data file binary or ascii*
  - *skip: number of lines to ignore before the start of the data*
  - *stride: how many columns to jump to the next item*
  - *offset: number of columns to jump over before the first element is reached*
- In the case of ASCII the steps are in delimited items

Example 1: Image Data

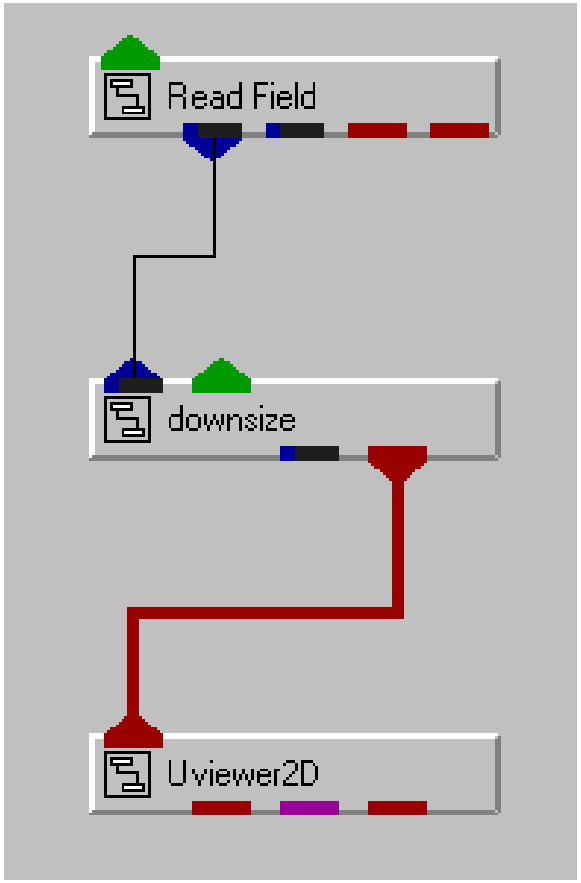
- MRI scan of the human brain
- The 2D data consists of:
  - 512x512
  - each data value is a single scalar byte
  - regular array of data
  - binary format with no header information
- What does the field file look like?

The Field file for Example 1: MRI.fld

```
# AVS field file
# MRI brain scan
ndim=2
dim1=512
dim2=512
nspace=2
veclen=1
data=byte
field=uniform
variable 1 file=mri.dat filetype=binary skip=0  stride=1
```

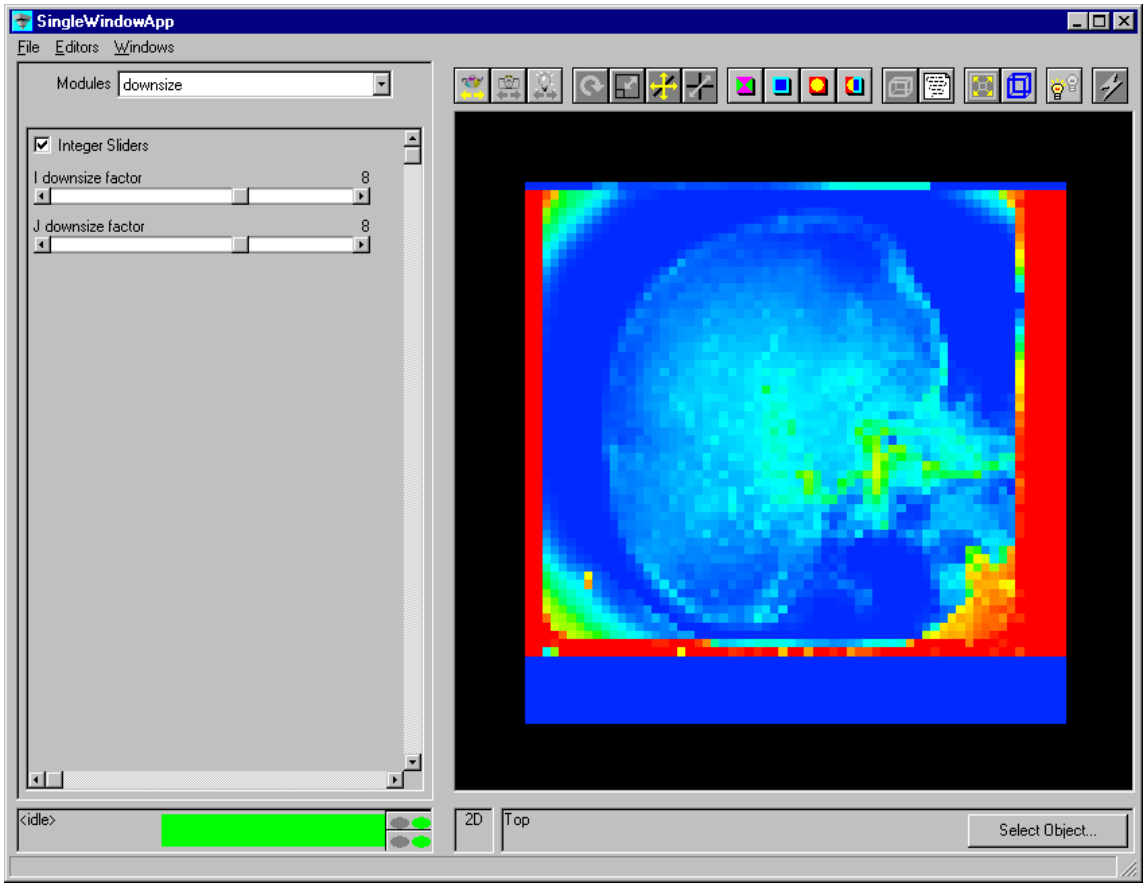
Importing the MRI Data

- Read_Field: reads an AVS field file and converts to AVS/Express field
- downsize: sub-samples the field
- Uviewer2D: creates a 2D image of the field in the DataViewer



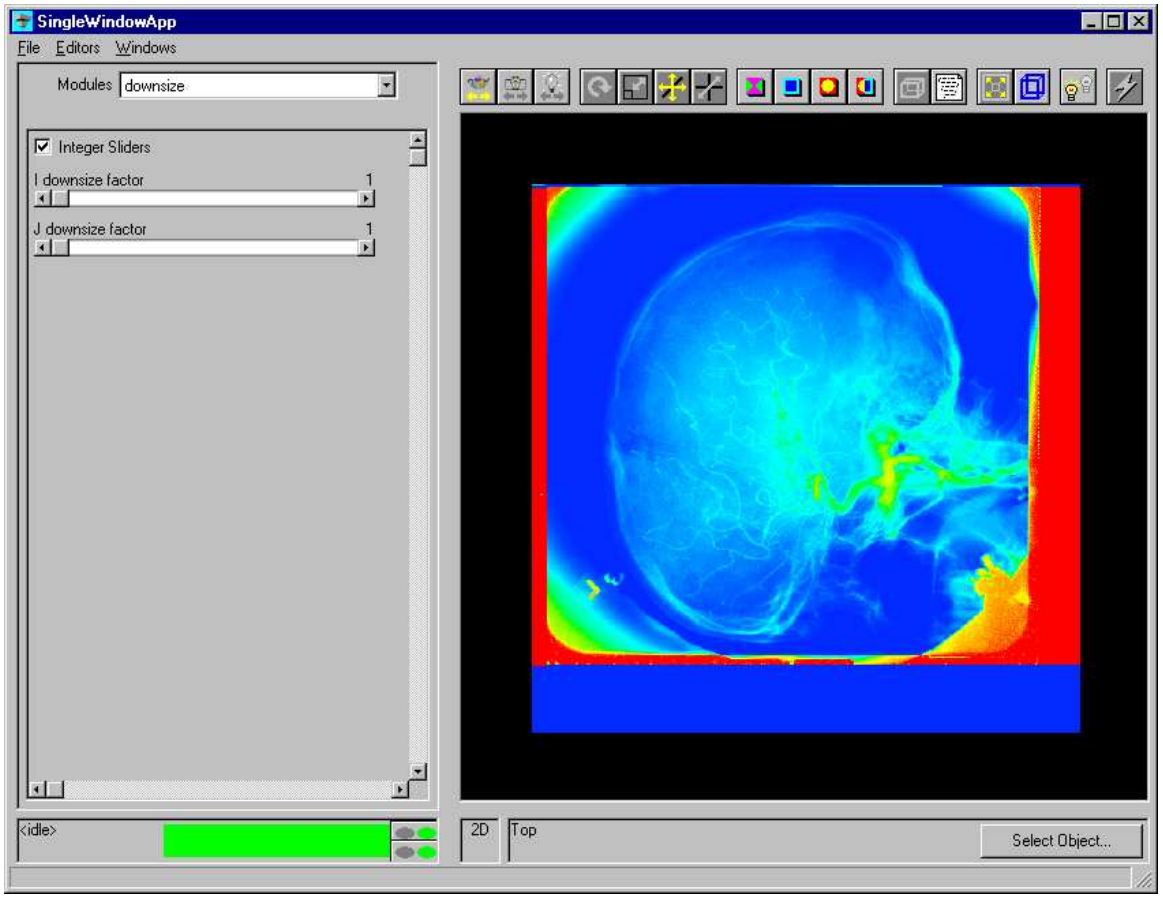
Importing the MRI Data

- MRI data as viewed with example application, using default downsize settings.

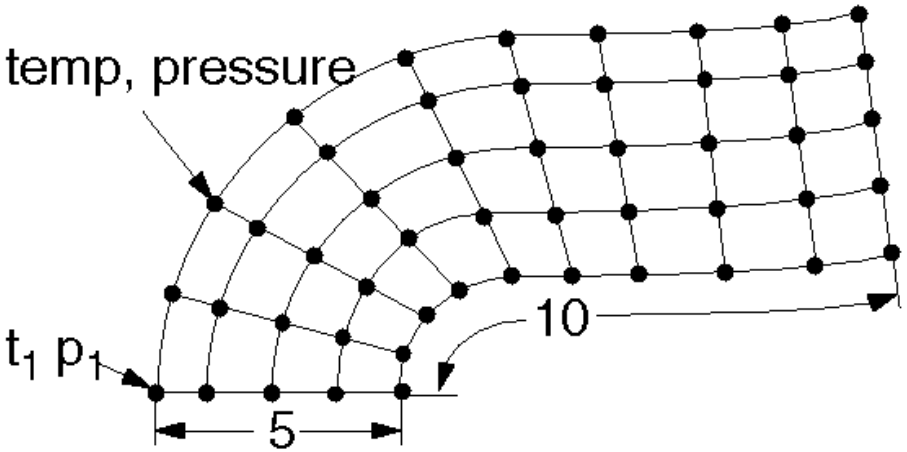


Down-sizing the Data

- Shown with all the imported data.



Example 3: Irregular Data



flow.dat

t_1 p_1 t_2 p_2 t_3 p_3 ...
...

coord.dat

x_1 y_1 x_2 y_2 x_3 y_3 ...
...

Solution to Example 3

```
# AVS field file
#
ndim = 2
dim1 = 5
dim2 = 10
nspace = 2
veclen = 2
data = float
field = irregular
label = temperature pressure
variable 1 file = flow.dat filetype = ascii skip = 0 offset = 0 stride = 2
variable 2 file = flow.dat filetype = ascii skip = 0 offset = 1 stride = 2
coord 1 file = coord.dat filetype = ascii skip = 0 offset = 0 stride = 2
coord 2 file = coord.dat filetype = ascii skip = 0 offset = 1 stride = 2
```

MANCHESTER 1824

The University of Manchester

3D Visualization Techniques

3D Example

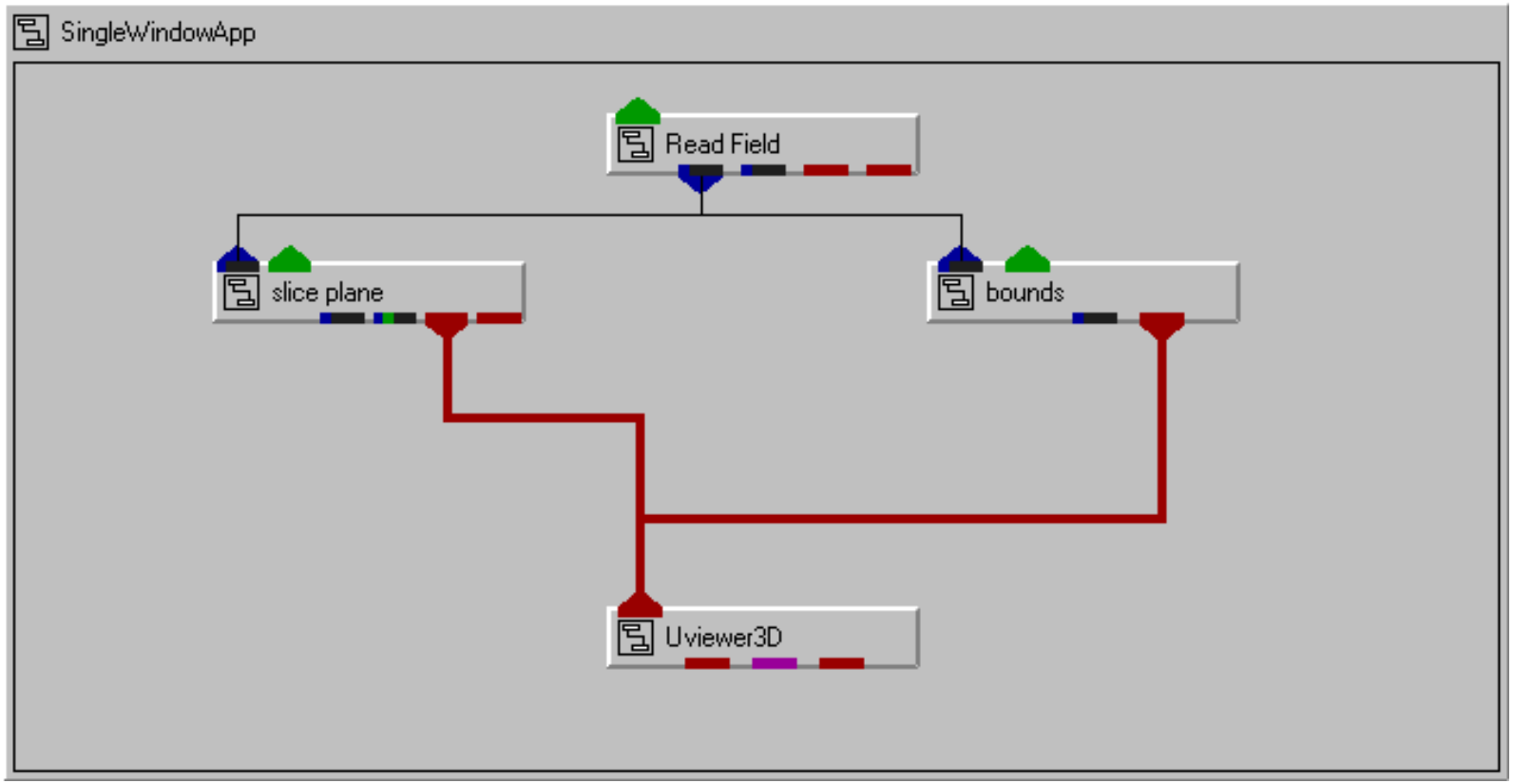
- There are 5 different components and associated 3D coordinate information
- 3 of the scalar components combine to form a vector
  - momentum vector
- Data file format (ASCII)
  - density x-momentum y-momentum z-momentum stagnation x y z
- Data on a 10x8x8 irregular grid

FIN.fld

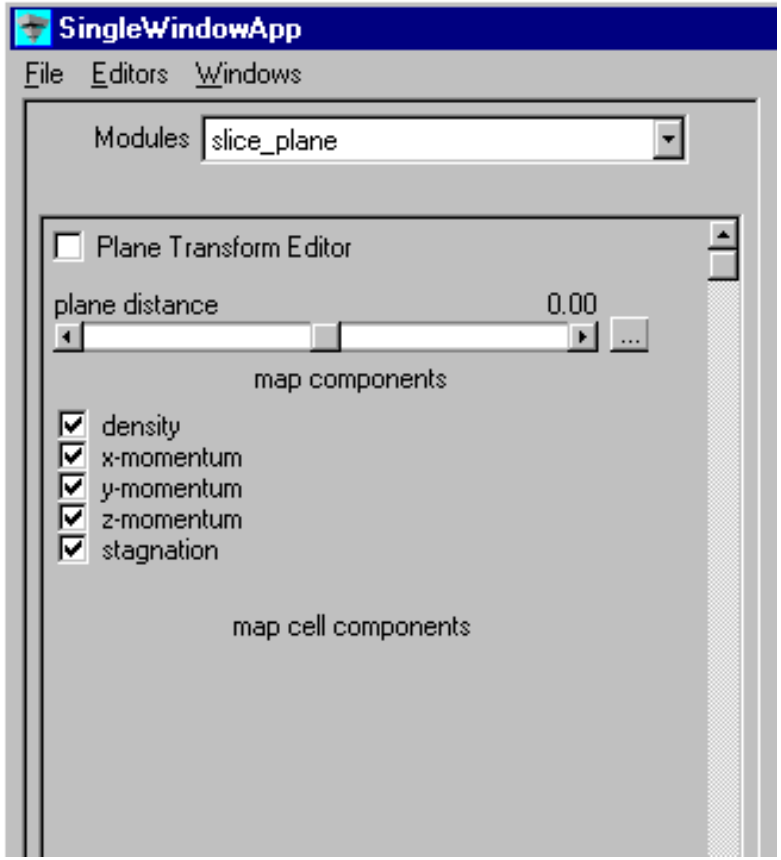
Flow of air over a blunt fin

```
# AVS field file
# Bluntfin
ndim = 3
dim1 = 10
dim2 = 8
dim3 = 8
nspace = 3
veclen = 5
data = float
field = irregular
label = density x-momentum y-momentum z-momentum stagnation
variable 1 file=fin.dat filetype = ascii skip = 0 offset = 0 stride = 8
variable 2 file=fin.dat filetype = ascii skip = 0 offset = 1 stride = 8
variable 3 file=fin.dat filetype = ascii skip = 0 offset = 2 stride = 8
variable 4 file=fin.dat filetype = ascii skip = 0 offset = 3 stride = 8
variable 5 file=fin.dat filetype = ascii skip = 0 offset = 4 stride = 8
coord 1 file=fin.dat filetype= ascii skip = 0 offset = 5 stride = 8
coord 2 file=fin.dat filetype= ascii skip = 0 offset = 6 stride = 8
coord 3 file=fin.dat filetype= ascii skip = 0 offset = 7 stride = 8
```

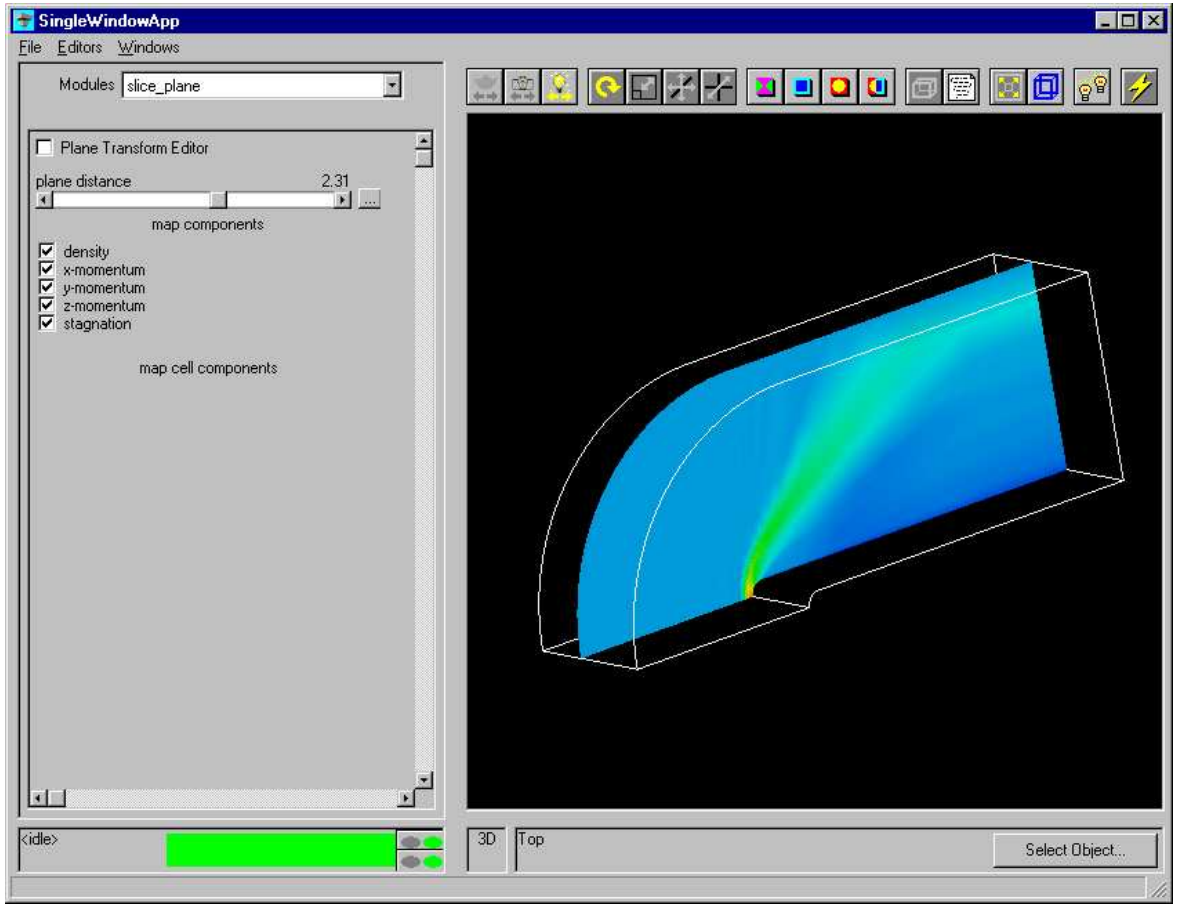
Visualizing Scalar Data



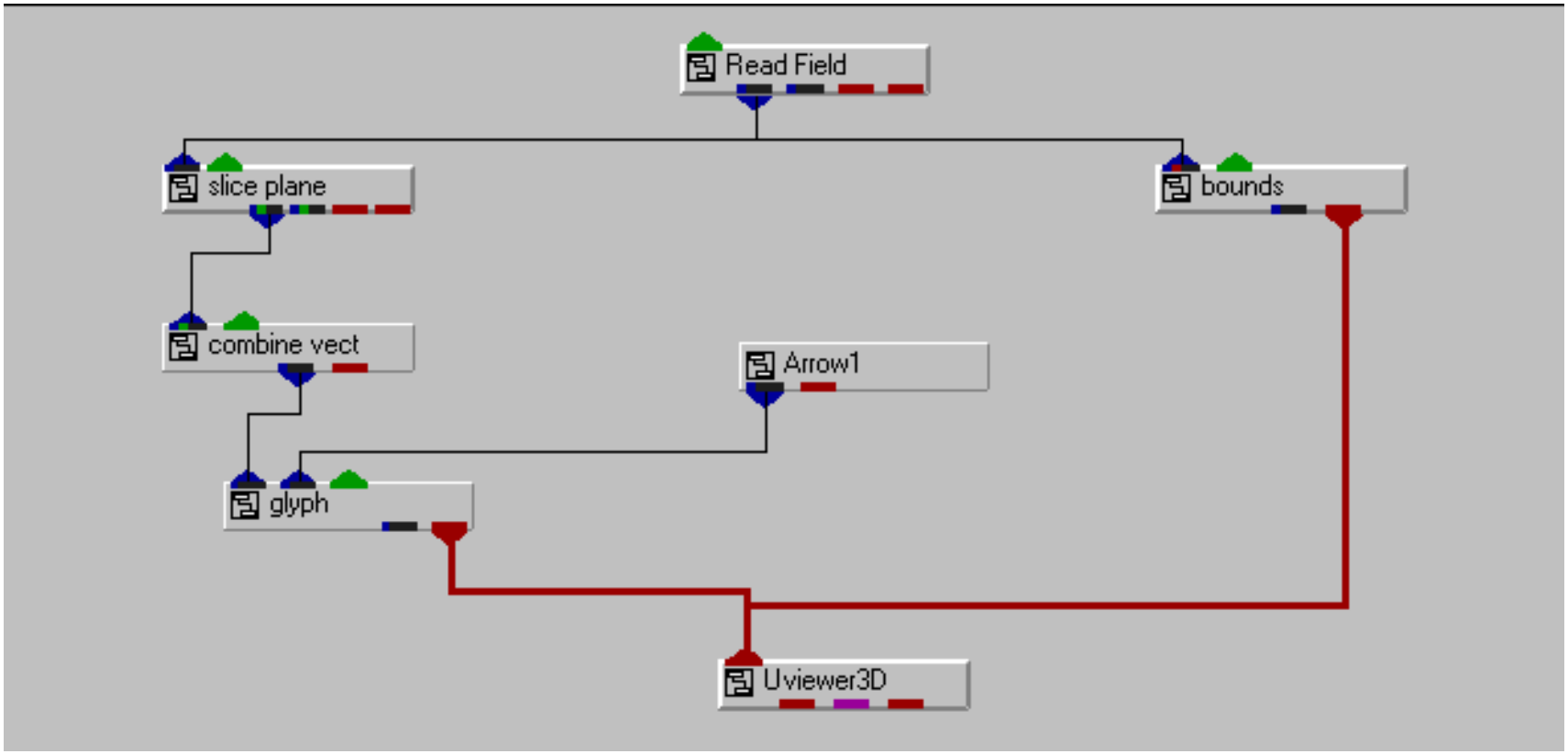
slice_plane


SingleWindowApp
File
Editors
Windows
Modules
slice_plane
Plane Transform Editor
plane distance
0.00
...
map components
density
x-momentum
y-momentum
z-momentum
stagnation
map cell components

slice_plane


SingleWindowApp
File Editors Windows
Modules slice_plane
Plane Transform Editor
plane distance 2.31
map components
density
x-momentum
y-momentum
z-momentum
stagnation
map cell components
<idle>
3D
Top
Select Object...

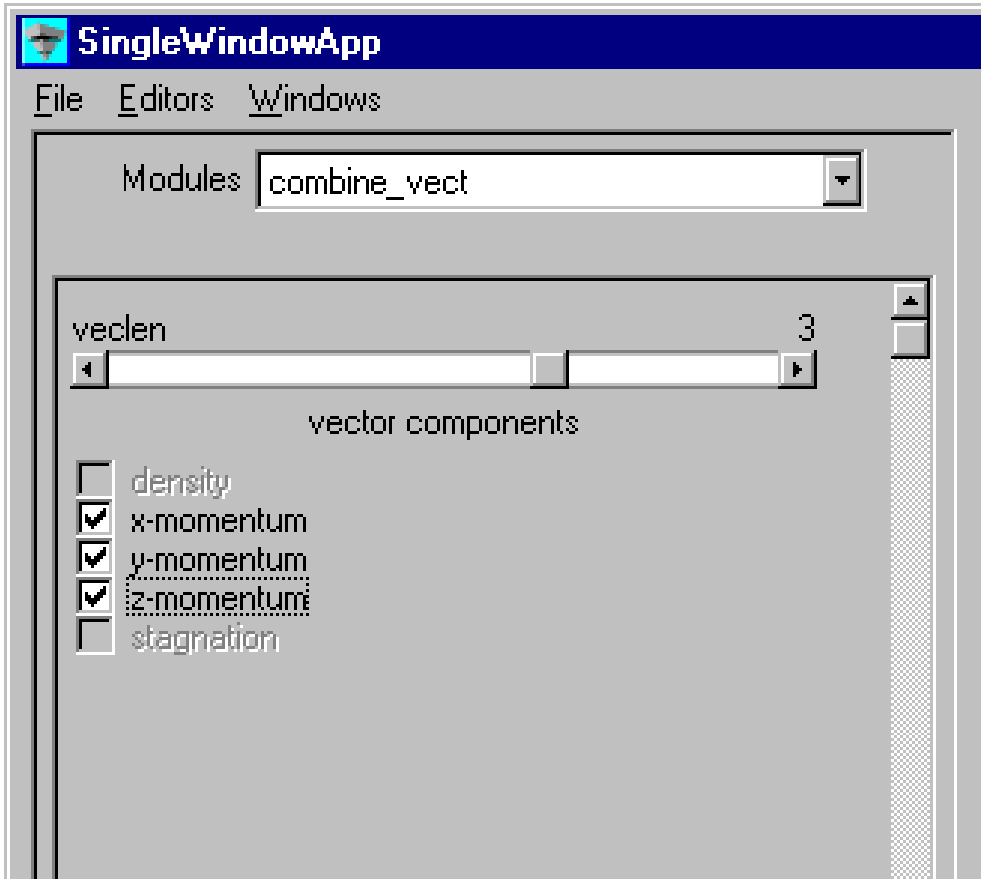
Visualizing Vector Data



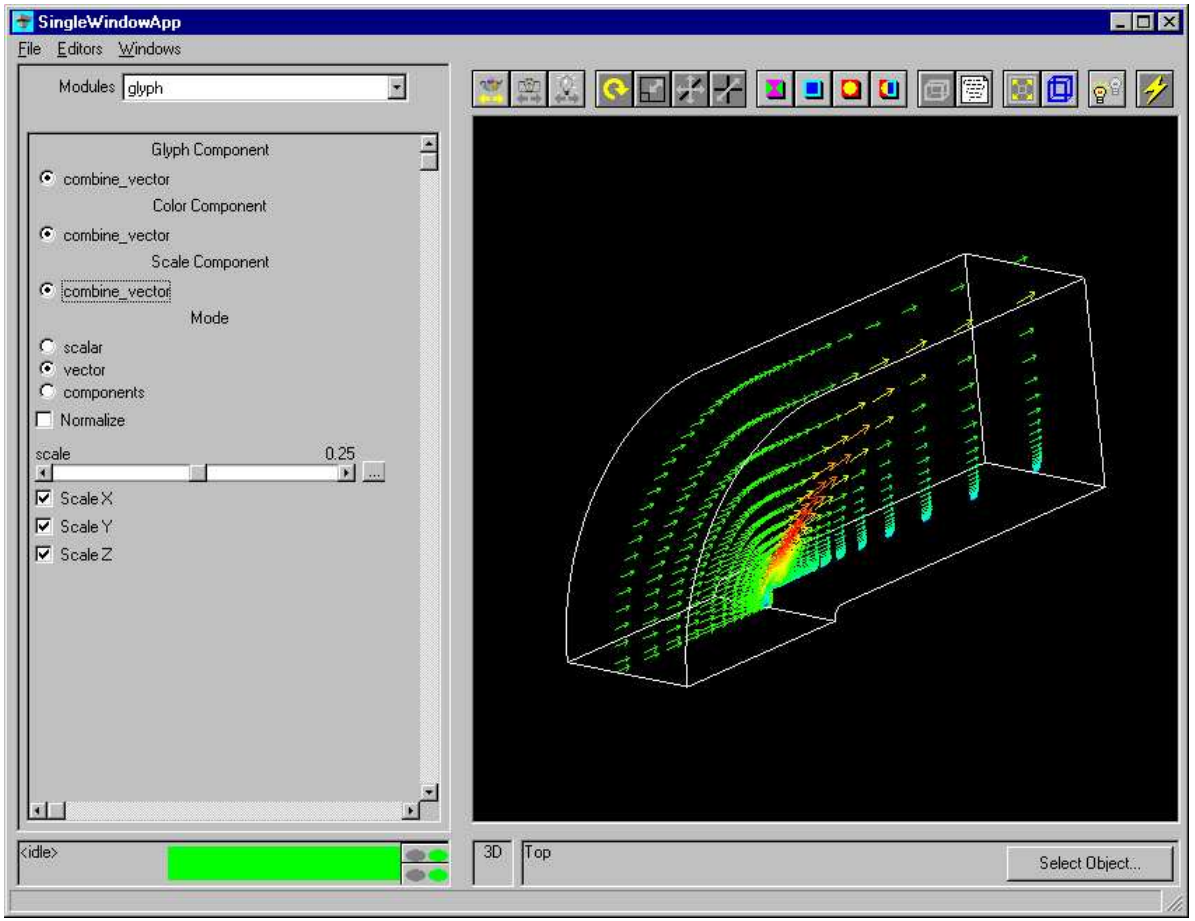
Visualizing Vector Data

- Slice_plane
  - *extracts arbitrary 2D slice from 3D field*
- combine_vect
  - *selects some or all components of 3D field*
- glyph
  - *creates geometric object indicating field values*
  - *uses Arrow1*
- bounds
  - *generates a bounding frame of the 3D field*

combine_vect



Glyph and Arrow1



Further Information

- AVS/Express documentation set
- Examples Library
- xp_demo on the AVS/Express CD
- Research Computing Services
  - http://www.rcs.manchester.ac.uk/
- International AVS Centre:
  - avs@iavsc.org
  - http://www.iavsc.org/