





## Development examples

1. Ferrari Racing Team – distributed CFD data access
2. MOD3D – 3D reconstruction from images
3. Elasis (Fiat Auto) – immersive CFD visualization
4. STM4 – Molecular visualization toolkit
5. Pelton Turbine project

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---



---



---



---



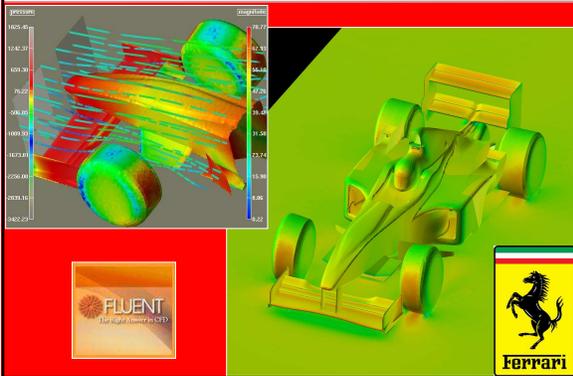
---



---



## Ferrari Racing Team




---



---



---



---



---



---



## Ferrari Racing Team

### Highlights:

- Standard visualization techniques + some data cut/optimization modules added
- Fluent reader added to AVS/Express
- Big data sets
- Performance were critical
- The prototype benefited by quick prototyping capabilities offered by AVS/Express (unfortunately the project stopped at the prototype phase...)

---



---



---



---



---



---




---



---



---



---



---



---



---

**MOD3D**

Highlights:

- Almost all was custom code
- GUI capabilities were critical (Data Viewer not used)
- Few visualization technique used
- Added modules for zooming, marking and navigation
- More details on: <http://www.cscs.ch/~mvalle/MOD3D>

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---



---



---



---



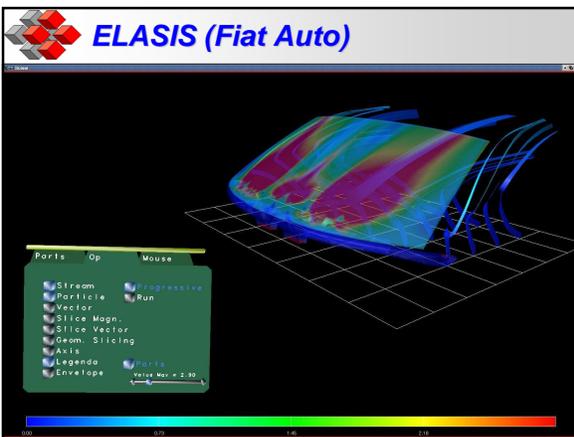
---



---



---




---



---



---



---



---



---



---



## ELASIS (Fiat Auto)

- Highlights:
- Standard CFD techniques
  - Input already in AVS UCD format
  - Use immersive Multi Pipe Edition (with wand and head tracker)
  - 3D widgets usable also from standard screen

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

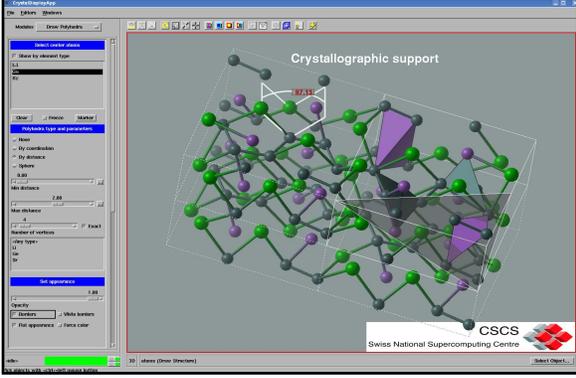
---

---

---



## STM4




---

---

---

---

---

---

---

---



## STM4

**STM4 will be the theme of this afternoon session!**

- Highlights:
- Library of components plus two new data types
  - Benefits from module architecture
  - Decoupling of AVS/Express interface from code through an interface library

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

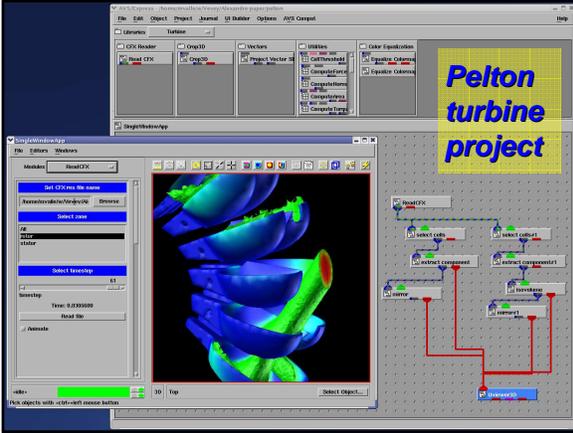
---

---

---

---

---




---

---

---

---

---

---

---

---

### Pelton turbine project

Highlights:

- Added CFX reader
- Added specialized visualization techniques
- Added modules to overcome AVS/Express annoyances (like extents on select cell set)
- Used a lot of validation modules (like computing integrals and surface areas)
- In depth study of colormaps that helps perception of fine details
- AVS/Express was mostly a prototyping environment

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

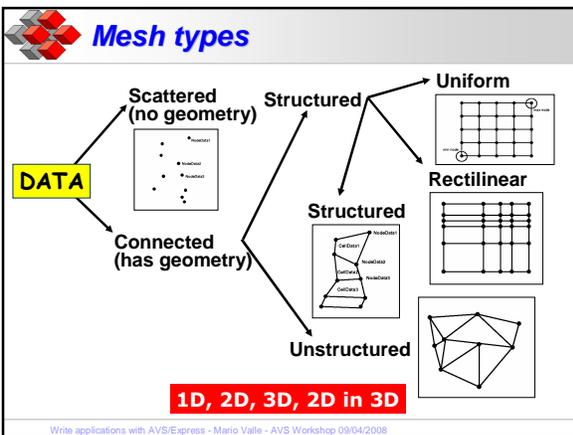
---

---

---

---

---




---

---

---

---

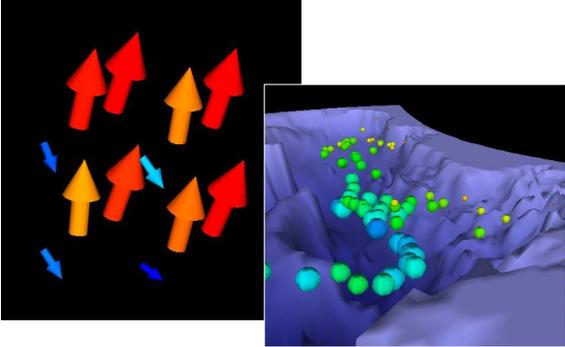
---

---

---

---

 **Scatter data**



Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

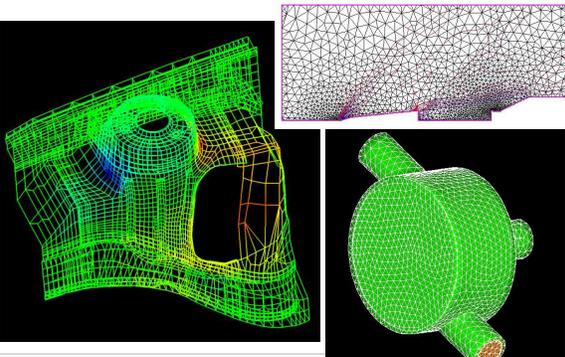
---

---

---

---

 **Unstructured grids**



Write applications with AVS/Express - Mario Valle - AVS Workshop

---

---

---

---

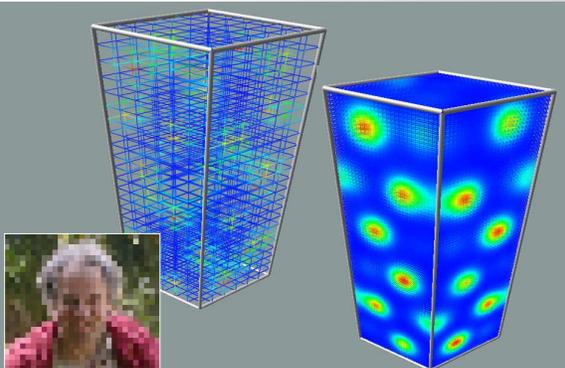
---

---

---

---

 **Uniform grids**



---

---

---

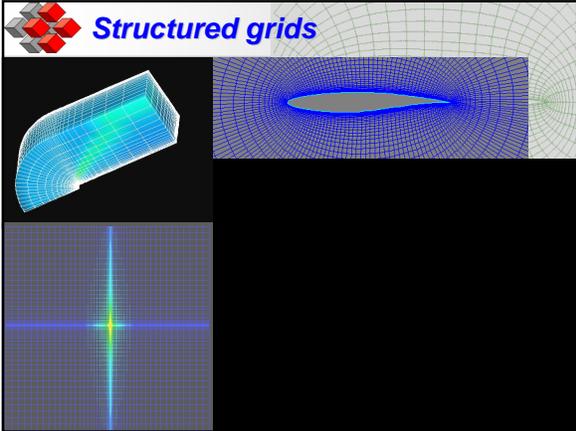
---

---

---

---

---




---

---

---

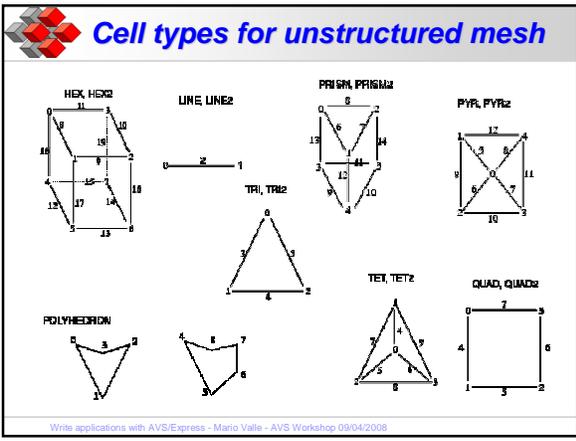
---

---

---

---

---




---

---

---

---

---

---

---

---

**Data import in AVS/Express**

- Various formats directly readable
- Rd\_Txt\_Columns
- Database (ODBC / Oracle)
- AVS Field file format
- File import modules
- Custom readers

The image shows a screenshot of the AVS/Express software interface. On the left is a list of data sources, and on the right is a list of data import modules including:
 

- (Read Field)
- (Read UCD)
- (Read Image)
- (Read Hdf5 Field)
- (Rd netCDF File)
- (Rd Txt Columns)
- (Rd Txt Grid)
- (Rd Txt Sequence)
- (Rd Bin Sequence)
- (Read DVF)
- (Read Polygon)
- (Read Triangle)
- (Read img2Vol)
- (Read Volume)
- (Read PLOT 3D)
- (Read CGNS)
- (Read Geom)
- (Read Geom)
- (Print Field)
- (Print Image)

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---

### Most important user data import

AVS Field file format

Rd\_Txt\_Columns

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---

---

---

### Import choices

<p>Simple</p> <p>Hard</p>	Various directly readable formats	Simplest solution
	Rd_Txt_Columns	Textual file. OK for scatter or Line meshes
	Database (ODBC and Oracle)	Read a table from the database
	AVS Field file format	Convert data to FLD. Support uniform, rectilinear and structured meshes
	File import modules	Read whatever you want and then combine with field mappers
Custom readers	No constrain, but need to program and know the FLD API	

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---

---

---

### Formats directly supported

- AVS field
  - Uniform, rectilinear, structured
- AVS UCD
  - Unstructured
- HDF5, Plot3D, netCDF, CGNF
  - Standard CFD formats
- Text
  - Need a table\_to\_scatter/table\_to\_uniform module
- DXF, Polygon, Triangle, AVS Geom
  - Geometries
- Images
  - BMP, GIF, JPEG, PBM, SGI RGB, TIFF

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

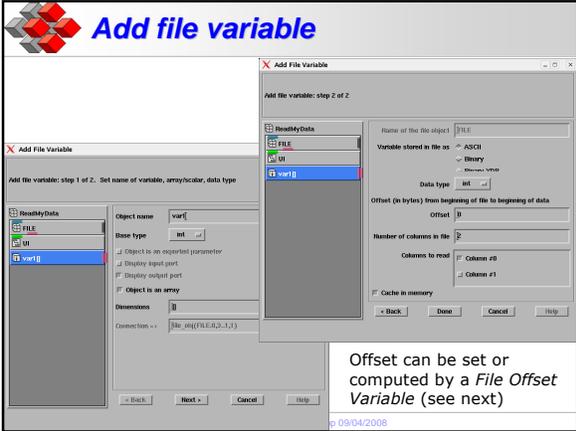
---

---

---

---






---

---

---

---

---

---

---

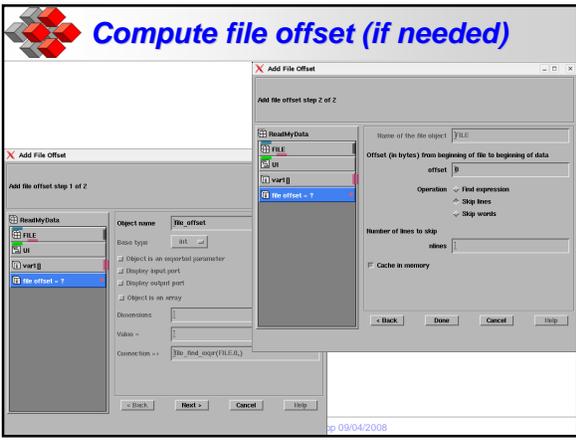
---

---

---

---

---




---

---

---

---

---

---

---

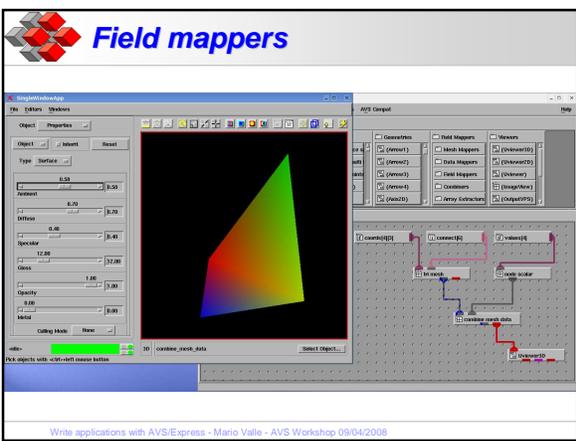
---

---

---

---

---




---

---

---

---

---

---

---

---

---

---

---

---



**Exercise**

1000				
18.37	5.00	10.00		.06
18.94	5.00	10.00		.07
19.49	5.00	10.00		.05

- Import file `conc.dat` using `Rd_Txt_Columns` (skip 1 line)
- Transform it into a scatter field using `table_to_scatter_field` (columns: `x`, `y`, `z`, `value`)
- Bonus: `glyph+Diamond3D` (normalize)

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



**Exercise**

- Read the AVS Field format file `splash.fld` (open it to see the format)
- Create a surface plot using `surf_plot`
- Change the vertical scale
- Add bidirectional light
- Change datamap (bonus: `ColormapEditor`)
- Add Legend

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



**Exercise**

- Read the AVS UCD format file `example2.inp`
- Build vectors on the nodes: `combine_vect`
- Glyph + `Arrow1` (scale  $1e-5$ )
- Offset to deform the cube
- magnitude to color the deformed cube

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



**Exercise**

- Read the AVS Field format file `hydrogen.fld`
- Extract a slice using `orthoslice (axis 2)`
- Build a surface with `surf_plot`
- Read `mandrill.x` using `Read_Image`
- Spread it on the surface with `texture_mesh`

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



**Why an application?**

1. Because AVS/Express is a developer environment for visualization applications
2. Because the user needs an application to solve a specific problem
3. Because the user doesn't want to think

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



**AVS/Express application types**

1. Pre-built interconnection of modules.  
Modules that could be:
  1. Modified standard modules
  2. Modules that group other modules into macros (use DV\* modules)
  3. Modules that encapsulate V code directly
  4. Modules that integrate C / C++ / Fortran code
2. A toolkit. The application is done by the user composing its modules
3. Runtime application (it does not need an AVS/Express installation to work)

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---

## Two AVS/Express editions

- AVS/Express Developer Edition
  - No limitations
- AVS/Express Visualization Edition
  - Some modules not present
  - Some modules cannot be expanded
  - Cheaper

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---

## Some AVS/Express resources

- AVS homepage: <http://www.avsc.com/>
- AVS forum: <http://forum.avsc.com/>
- Official documentation: <http://help.avsc.com/Express/>
- AVS/Express built-in examples
- Visualization techniques manual
- International AVS Center (IAC): <http://www.iavsc.org/>
- IAC training material: <http://www.iavsc.org/training>
- Patches, doc and examples: <ftp://ftp.avsc.com/pub/>
- Customer projects: <http://www.iavsc.org/general/links/>
- Other links on: <http://www.cscs.ch/~mvalle/AVS/>

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---

## AVS/Express project structure

The diagram illustrates the project structure. At the top, a blue line labeled 'Sources' has arrows pointing to a row of boxes: avscenv, v/, bin/, lib/, include/, bid/, and express.\*. Below this row, a 'pj' box is connected to the 'bin/' box. Under 'v/' are 'templ.v' and 'pc/'. Under 'pc/' is 'express.exe'. An arrow points from 'templ.v' to 'avscenv' with the label 'XP\_PATH definition'. Another arrow points from 'pc/' to 'pj' with the label 'Save Project - Workspace content'. A note 'This is the MACHINE value' points to the 'pc/' box. Below 'bid/' is a 'module1/' box. The 'express.\*' box contains the text: 'express.\*', 'xexpress.\*', 'user.\*', 'xuser.\*'.

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

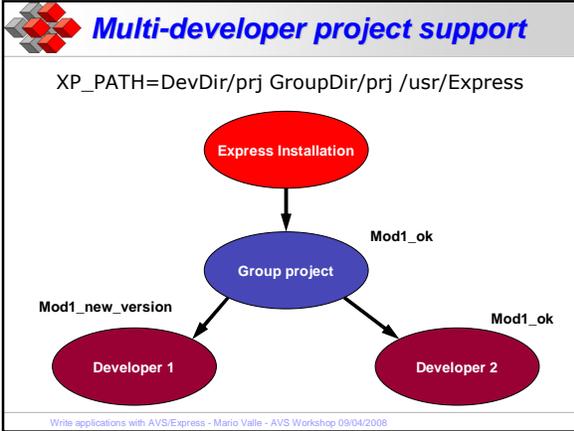
---

---

---

---






---

---

---

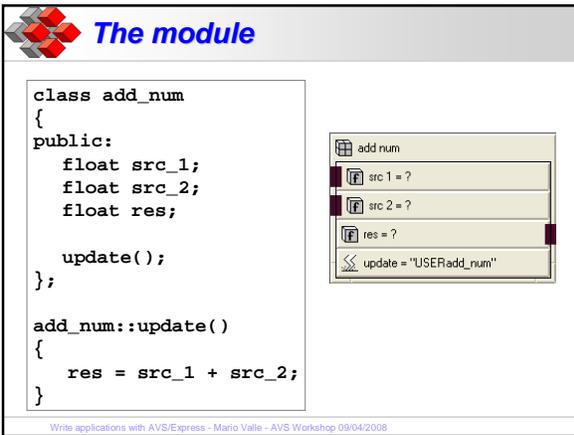
---

---

---

---

---




---

---

---

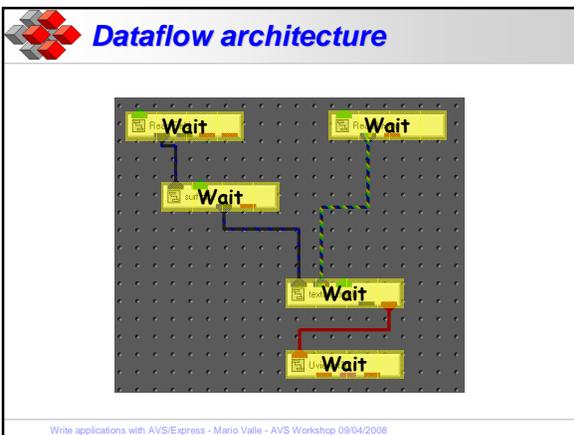
---

---

---

---

---




---

---

---

---

---

---

---

---

**Dataflow architecture**

Enter filename

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---

**Dataflow architecture**

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---

**Dataflow architecture**

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

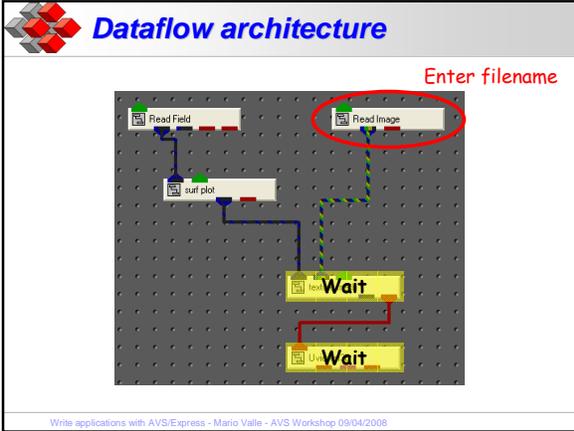
---

---

---

---

---




---

---

---

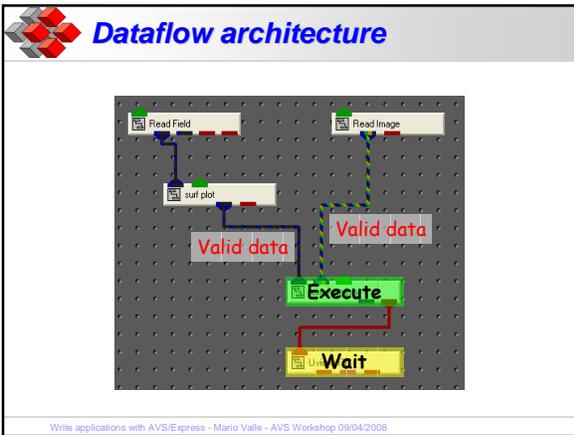
---

---

---

---

---




---

---

---

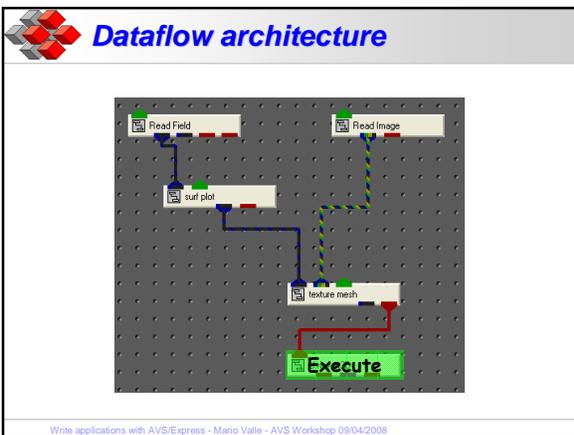
---

---

---

---

---




---

---

---

---

---

---

---

---

### Param/method interaction

The screenshot shows the 'Object Editor' window with the 'Parameter' editor selected. The 'Methods' list contains 'match update'. Below, the 'Settings for parameter/method interaction' section has four checkboxes: 'notify' (unchecked), 'read' (unchecked), 'write' (unchecked), and 'req' (unchecked).

**read, write:** usage direction by noted method  
**notify, req:** declare when noted method fires

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---

### More complex module

The screenshot shows a complex module with many methods and attributes. The methods list includes 'UpdateIndicators', 'UpdateTableFromValues', 'UpdateValuesFromTable', 'bin', 'natoms = 0', 'atoms ikk', 'render indicator = "normal"', 'render info', 'natoms = ?', 'render()', 'nbonds = ?', 'bonds type()', and 'bout'.

- More than one method
  - Each triggered by a different set of parameters
- Reference modes:
  - By value (^)
  - By reference (&)
  - By pointer (\*)
- Data types:
  - Atomic: int, float, byte, enum
  - Arrays
  - Groups

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---

### on instance / on deinstance

The screenshot shows the 'Add Method' dialog box. The 'Object name' is 'NewMethod'. The 'Method type' is 'C++ (comethod)'. There are two checkboxes: 'Method runs when object is instantiated' (unchecked) and 'Method runs when object is deinstantiated' (unchecked). The 'Weight' is set to '1'. Buttons for '< Back', 'Done', 'Cancel', and 'Help' are visible.

Equivalent to C++ constructor / destructor

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---

## Steps to add a computing module

1. Create a project
2. Start the Add Module Wizard
3. Remember to do a Save Project!
4. Add your code to the skeleton
5. Compile (`express.dsw`, `express.sln` or `make -f express.mk`)
6. Exec the new `bin/pc/express` or `bin/linux/express`

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

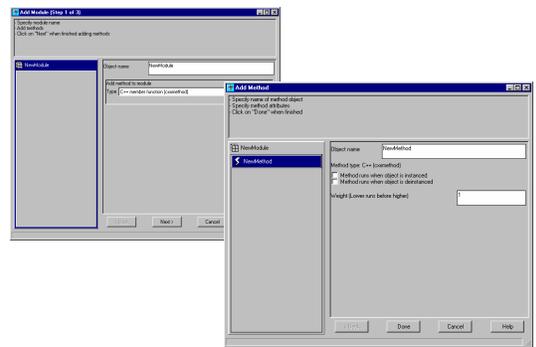
---

---

---

---

## Add Module Wizard (1 of 3)



Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

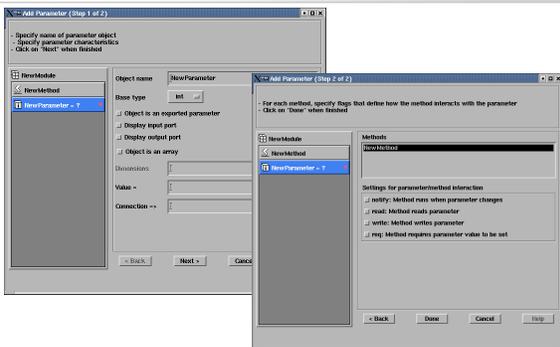
---

---

---

---

## Add Module Wizard (2 of 3)



Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---

### Add Module Wizard (3 of 3)

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

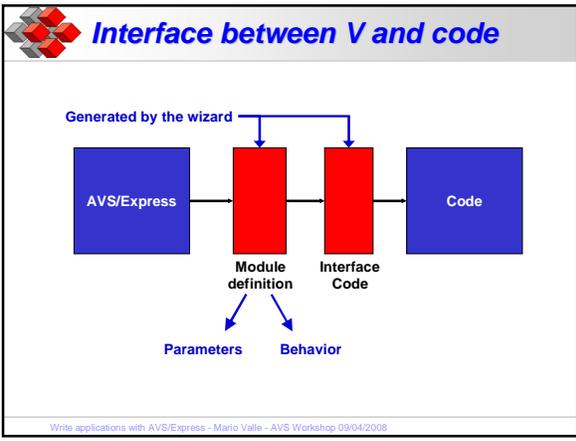
---

---

---

---

---




---

---

---

---

---

---

---

---

---

---

### Beware!

**Save Application  $\neq$  Save Project**

**Use Workspaces**

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---

---

---



## Exercise

- Create a module `MyAdd` in C (two float `p1` and `p2` and float output `out`)
- Create a module `MyAdd1` in C++ (two float `p1` and `p2` and float output `out`)
- Test it
- Find the differences between the two languages

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## C – C++ – Fortran differences

- Same philosophy for C and Fortran
- C++ is more similar to the Express structure: every module is a class
- In C++ interfacing to OM is hidden to the user
  - In the course we will use C++ only (also as a better C)
- When is C++ not suited? Some field creation operations, some OM operations.
  - You can always call OM or FLD routine directly (using a small trick)

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## Another module creation method

- Use a text editor
- Create a `YourModule_mod.v` file describing the module interface
- Add it to `templ.v` (for now using `$include`)
- The module will appear in Libraries → Templates
- Now we look at a module definition example, then we introduce the V language.

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## Another module creation method

```

module BackboneCore<src_file="Backbone.cxx",
  out_hdr_file="Backbone_gen.h",
  out_src_file="Backbone_gen.cxx",
  cxx_hdr_files="../types/MoleculeType.h",
  build_dir="cscs_proj/stm4/tube"> {

  cxxmethod+req DrawTube(
    .molecule.xyz_lst+read+notify+req,
    .molecule.atom_name+read+notify+req,
    .tube_coordinates+write
  );

  CSCS_PROJ.STM4.TYPE.MolecularType
    &molecule<NEportLevels={2,0}>;
  float tube_coordinates<NEportLevels={0,2}>[];
};

```

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---

---

---



## The V language

- It is a declarative language
  - No loops, conditionals, call, etc.
- Describes (almost) exactly all that can be done graphically
- Describes a hierarchy of objects and connections between them
- V code could be entered:
  - From the prompt (called VCP)
  - In a text file to be read inside Express
  - Saving and modifying an existing application
- V has some construct not strictly related to the network (\$-command, <...> properties, built-ins)

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---

---

---



## ➔ Exercise ➔

- Create a network with the former **MyAdd1** module
- Save the application into **ex2.v**
- Edit the **ex2.v** file and look at the generated V code

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---

---

---



## Object hierarchy

- The nesting of modules

```
{
  {
    };
};
```

- Connections

```
obj1 => obj2; (it is different from obj1 = obj2?)
```

- Moving along the hierarchy

```
<-
Obj1.subobj1.subsubobj1
```

- Creating / deleting objects

```
Type new_obj; (eg. int myint);
-new_obj;
```

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## Unusual constructs

- V make simple dynamic structures:

```
group A {
  int len;
  float array[.len];
};
```

- V build-ins work on whole arrays:

```
float x[5] => init_array(5, 1, 5);
float res[5] => log(x);
```

- Built-ins to interrogate the hierarchy:

```
group my_group[6] {
  int idx => index_of(my_group);
  string name => name_of(<-, 1);
};
```

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## \$-commands

### \$list

List an object content

### \$print

Print an object value

### \$int, \$float, etc.

Convert the value

### \$get\_array

Print the content of an array

### \$echo

To monitor V file loading

### \$help

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## Exercise

- From the VCP prompt create an integer variable
- Assign it a value
- Create a second one connected to the first one
- Delete the first integer

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## Exercise

- Create a network with the former `MyAdd1` module
- Save the application into `ex2.v`
- Edit the `ex2.v` file and replace the module with a sum computed in V

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## V data types

- Atomic data types:
  - int, float, string, enum (!), prim
- Arrays
  - byte pixels[768][576];
  - string names[.len];
  - Beware: access an array entry by: !pixels[12][128] = 5;
- Aggregated data types (structures):
  - group
- User defined data types
  - Whatever you want

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## Exercise

- Create a group, call it **MyType**, and populate it
- Pretend it is your data type
- Create a variable of type **MyType**

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

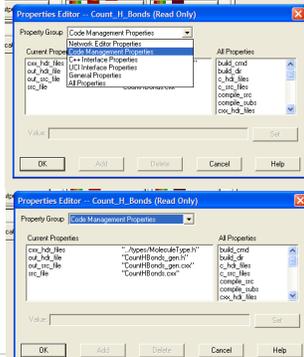
---

---



## Properties

- < ... >
- Property Editor
- NE property group (NEX, NEy, NEportLevels)
- Code Management property group (src\_file)
- Specific properties for C++, C



Write applications with AVS/Express - Mario Valle -

---

---

---

---

---

---

---

---



## V code organization

- Workspaces
  - They are simply places where I can create my modules
- Templates
  - templ.v (Remember project structure?)
- Libraries
  - Organize the access to the Template modules

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## Visible access to modules

- `lib_xp.v` (`lib_vxp.v`)  
(Re)defines the Libraries content adding links (using NElink)
- Example:

```
*$XP_PATH<0>/v/lib_xp.v Libraries {
  library STM4 <NEdisplayMode="opened",NEhelpFile="STM4/index.html"> {
    NElink Readers_writers <NEdisplayMode="open"> => Templates.CSCS_PROJ.STM4.LIB.DATAIO;
    NElink Bonds <NEdisplayMode="open"> => Templates.CSCS_PROJ.STM4.LIB.BONDS;
    NElink Modules <NEdisplayMode="open"> => Templates.CSCS_PROJ.STM4.LIB.MODULES;
    NElink Full_Apps <NEdisplayMode="open"> => Templates.CSCS_PROJ.STM4.LIB.APPS;
    NElink Experimental => Templates.CSCS_PROJ.STM4.LIB.EXPERIMENTAL;
  };
};
```

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## V for gurus

- `+nres`
  - If you have undefined errors when loading libraries
- `+notify +noreq`
  - For triggering problems
- `merge()`
  - To substitute part of a structure
- `$match`
- `#ifdef`
- `$include / $resolve / $XP_VPATH`

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## merge()

<pre>group original {   ...   int other = 99;   float value = 7.3;   ... };</pre>	<pre>group substituted {   float value = 1.2; };</pre>
-----------------------------------------------------------------------------------	--------------------------------------------------------

```
group &modified => merge(substituted, original);
.....>
```

<code>\$float modified.value</code>	<code>\$int modified.other</code>
1.200000	99
<code>\$float original.value</code>	<code>\$int original.other</code>
7.300000	99

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## Exercise

- Create an empty application
- Add to it the previous example
- What happens modifying the two values?

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## Module programming

"Put flesh around the AVS/Express generated skeleton"

1. Argument passing (C++ is simpler)
2. FLD reading/writing from a module
3. Use OM and FLD C routines from C++
4. Accessing OM flow control from C++

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## Argument passing

- Atomic data
 

```
module_param = local_var;
local_var = module_param;
```
- Array (do not forget ARRfree(!))
 

```
float *xyz_arr =
  (float *)module_array.ret_array_ptr(OM_GET_ARRAY_WR);
xyz_arr[123] = 45.6;
ARRfree(xyz_arr);
```
- To read use OM\_GET\_ARRAY\_RD, to modify use OM\_GET\_ARRAY\_RW

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## Argument passing

### Set array size

```
module_param.set_array_size(123);  
Or directly from V
```

### Set multi dimensional array sizes (eg. labels\_pt[1][3] )

```
int d[2];  
d[1] = 1;  
d[0] = 3;  
OMset_array_dims(labels_pt.obj_id(OM_OBJ_RW), 2, d);
```

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## Argument passing

### Get array size

```
len = module_param.ret_array_size();  
int len;  
float *xyz_arr = (float *)  
module_array.ret_array_ptr(OM_GET_ARRAY_RD, &len);
```

### Get multidimensional array sizes

```
int ndims;  
int dim_sizes[OM_ARRAY_MAXDIM];  
OMget_array_dims(my_array.obj_id(OM_OBJ_RW),  
                &ndims,  
                &dim_sizes);
```

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## Argument passing

### groups

- Simply use structure notation: mygroup.param = 12;

### string arrays

```
atom_labels.set_str_array_val(idx, "label");  
char buffer[256];  
char *s = in.atom_name.ret_str_array_val(idx, buffer);
```

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## Field read/write

- Use the provided C++ skeleton
- Import/export arrays and use combiner modules
- Use FLD routines for specialized functions
 

```
FLDset_cell_data_type(out_density.cell_set.obj_id(OM_OBJ_
RW), 0, DTYPE_FLOAT);
FLDget_node_data_ncomp(in.obj_id(OM_OBJ_RW),
&nnode_data);
```
- To pass the required OObj\_id use:
 

```
Object_in_module.obj_id(OM_OBJ_RW)
```

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## Access OM flow control

- To check who triggered the method
 

```
obj.changed(seq_num)
```
- To check if the object is valid (without using +req)
 

```
obj.valid_obj()
```
- To modify a parameter so it changes value immediately without waiting for module execution end
 

```
for(i=0; i < 1000; ++i)
{
  push_ctx(); out_value = i; pop_ctx();
}
```

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## Other useful routines

- Error messages
 

```
ERRverror()
```
- Confirmation dialogs
 

```
ERRsync_yesno_dialog()
```
- Update progress bar
 

```
OMstatus_check()
```
- Execute V code
 

```
OMparse_buffer(pair_list.obj_id(OM_OBJ_RW), "pair_list=>", 0);
```
- Make array really empty
 

```
obj.set_array(OM_TYPE_FLOAT, NULL, 0,
OM_SET_ARRAY_FREE);
```

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## Exercise

- Read a list of 3D points from a file
- Create a module that read them and interpolate a user given number of points between them
- What can I use to move a glyph along the interpolated path?

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## Organize your V code

- Module definitions  
Module\_mod.v
- Macros
- Applications
- Libraries ("file" lib, .vo files)
- The visible Libraries  
lib\_xp.v (lib\_vxp.v)
- Correlated properties  
compile\_subs

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## Application structure

- IAC projects structure  
(<http://www.iavsc.org/repository/pfiles/>)
- Divide Application / UI (UI => param, avoid the reverse)
- Low level modules / (medium) / end-user modules
- Parameter Blocks
- Static data

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## Code organization

- Build dir
- Naming conventions (\*\_gen, \*\_mod, \*\_struct)
- Use out\_hdr\_file / out\_src\_file
- cxx\_members to add methods
- cxx\_name if needed
- c\_src\_files / cxx\_src\_files etc. / link\_files
- c\_hdr\_files / cxx\_hdr\_files for class definitions

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## Suggested steps

- Create the base structure (templ.v, lib\_xp.v, libZZZ.v)
- Create object / test / Save Objects...
- Create modules using Wizard or cut&paste
- Use ScratchPad or directly edit files (Save Proj modifies the layout)
- Modify lib\_xp.v
- Base -gen\_proc express (script)
- Express.sln / make

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## Interesting API

- Asynchronous data handling (EVadd\_select())
- Dynamic object creation
- GEOM API

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## Interesting accessories

- **parse\_v**
  - to create connections, to instantiate objects
- **instancer**
- **copy\_on\_change**
  - Eg. for increment / decrement
- **shell\_command**
- **error\_handler**
- **Macro array**
  - `index_of()` / `name_of()`

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## Philosophy ...

- **Derive, not rewrite or modify**
- **Think about reuse**
  - You are creating components
- **Do not (re)invent every time!**
  - Use V if possible
- **Consider making it publicly available**

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## Testing / Debugging

- **V / OM level – Your code level**
- **Verbose (events) – modules that do not trigger**
- **Continuous build / multiplatform build**
- **If you are building runtimes, try building them as soon as possible**
- **`$valid $notify $refs_to`**
- **`$set_trace`**
- **Use debugger / valgrind**

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## Application performance

- Instancer (<instanced=0> + instancer module )
- Use menu Configure
- Use \$count\_obj and reduce object count
- Create "light" objects (Cross2D, point\_mesh, DataObject)
- Remove useless operations on viewers (like caching)
- Verify memory usage (set\_array / \$set\_array\_trace)
- Reduce data copy

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## Application performance (cont.)

- Orthoslice / TileRenderer
- Disable status / module flashing
- No autonormalize, no caching
- link instead of &group\_ref
- Define who allocate and who use the memory
- Use DV modules
- Uviewer 3D/2D
- Reduce cell sets count

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## Portability

- The whole world is not Windows
  - CamelCaseFileNames
  - Prefer "/" versus "\"
- Avoid absolute paths
  - Use getenv("PRJDIR") or \$ICONS
  - Use FILEmap\_variables("\$XP\_PATH<1>/data/atom-data.dat", file\_buf);
- Use #ifdef for V and C++
- Use relative dimensioning for GUI

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---

---

---

---

---

---

---

---



## Express Runtime creation

- Build & test (express process)
- Configure
- Save Compiled Project (selecting the application)
- Edit avsenv ( XP\_PATH=. )
- Create go.bat (env var + bin\pc\express -novcp)
- Clean RT directory / Add application data
- License and SET XP\_FEATURE=XP\_RUNTIME
- Test !!!! (add need\_obj, lib\_deps)

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---



---



---



---



---



---



---



## ARRIVEDERCI!

I hope you have enjoyed this introduction

Don't hesitate to contact me with your questions, curiosities and crazy ideas

(Remember, I'm here also this afternoon for STM4)



[www.cscs.ch/~mvalle](http://www.cscs.ch/~mvalle)  
[mvalle@cscs.ch](mailto:mvalle@cscs.ch)

Write applications with AVS/Express - Mario Valle - AVS Workshop 09/04/2008

---



---



---



---



---



---



---