Relational data input for PHOENICS

Relational input

PHOENICS User Meetings
PARIS, 2008

Contents

- The need for a **relational** input capability
- The **Advanced PHOENICS Input Language**
- The VR-Editor in **'protected mode'**
- **PRELUDE,** the **pre**-pre-processor
- The '**Gateway**' concept
- A room-fire **example**
- **PARSOL,** local **grid refinement** and **multi-runs**

By Brian Spalding, September, 2008

Relational data input to PHOENICS

Relational input

PHOENICS User Meetings
PARIS, 2008

Please note

This presentation has been prepared for persons who are already familiar with **PHOENICS**, and especially for users of its version for heating, ventilating, air-conditioning and fire simulation, **FLAIR**.

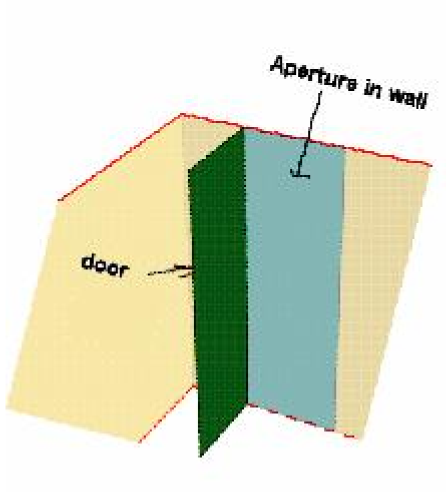
Persons more familiar with **other CFD codes** might care to ask themselves: *'Does my code have **relational data-input** capabilities? If so, how do they compare with those of PHOENICS? But, if not, why not?'*

The need for a relational input capability

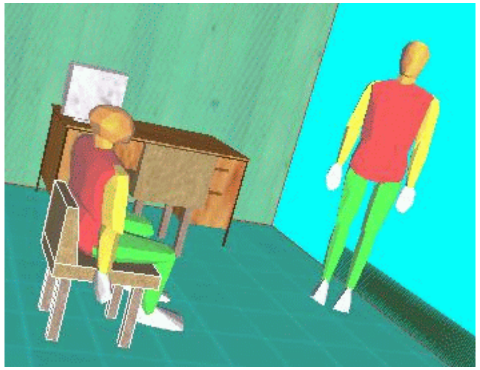
Relational input

PHOENICS User Meetings
PARIS, 2008

It is often required to ensure that the positions and sizes of objects conform to some **rules.** For example, doors must be of the right size **to fit** apertures in walls.



Similarly chairs must have their legs **in contact** with the floor; and sitting persons must be **in touch** with their seats.



Then if one **moves** the aperture or the chair, one needs the door and the person to **move with them**.

The need for a relational data-input capability

Relational input

PHOENICS User Meetings
PARIS, 2008

The PHOENICS Virtual-Reality Editor does have a '**grouping**' feature which enables **relative-position** connections to be expressed and recorded in the Q1 file; but it does **not allow** members of the group to **change relative size or position**.

Therefore, if the Q1 is to be used again with even slightly modified geometry, the user has to re-define the lost relationships **all over again**.

This deficiency has now been remedied, in two different ways: by

1. use of the VR-Editor in **'protected mode'**, and more fully
2. use of the new Graphical User Interface: **PRELUDE**.

The historical background:
the rise and **temporary** eclipse of the
PHOENICS Input Language

Relational input

PHOENICS User Meetings
PARIS, 2008

• In the early days of PHOENICS, data-input was effected by way of **assignment statements**, edited into files (Q1s). The statements were expressed in terms of the first **PHOENICS Input Language**, known as **PIL**.

• During the following years, PIL acquired many **new capabilities**: **Logic**al structures, **DO**-loops, capabilities in respect of **graphics**, **file-handling,** *etc*. This **'advanced PIL'** still flourishes; and it is used with much success by experts.

• Advanced PIL is **well able to express** the required **relationships** between the sizes and positions of different objects in a scenario.

• As the number of **new users** of PHOENICS increased, many of whom were **reluctant to learn** PIL, **menu-based** input procedures were provided: users **clicked buttons** or typed characters into boxes; then the PHOENICS Satellite **wrote the Q1 file for them**. Most users nowadays use these menus exclusively.

• **However,** although many advanced-PIL features are **exploited** by the menu system, they **do not appear in the Q1s** which the Satellite writes.

• Therefore, even if expert users hand-edited relationships into a Q1 file, once the VR-Editor had read them, it **recorded only their numerical implications**.

Examples of the obliterating tendency of the VR-Editor

Relational input

PHOENICS User Meetings PARIS, 2008

Example 1. An advanced-PIL expert might write:

```
REAL(width, height)                    ! declarations
width=0.85; height=1.80                ! settings
> OBJ, NAME, DOOR
> OBJ, SIZE,  width, 0.0, height       ! uses

> OBJ, NAME, APERTURE
> OBJ, SIZE, 0.0, width, height        ! uses
```

Having **read** the above, the VR-Editor would **write** simply:

```
> OBJ, NAME,       DOOR
> OBJ, SIZE,       0.850000E+00, 0.000000E+00, 1.800000E+00
> OBJ, NAME,       APERTURE
> OBJ, SIZE,       0.000000E+00, 0.850000E+00, 1.800000E+00
```

The Editor retains only the single-instance significance; but it **obliterates the declarations**.

Examples of the obliterating tendency of the VR-Editor

Relational input

PHOENICS User Meetings
PARIS, 2008

Example 2. An advanced-PIL expert might write:

```
REAL(size1, size2)                  ! declarations
size1=1.0; size2=2.0                ! settings

if (size1.gt.size2) then            ! condition
> OBJ, POSITION, 0., 0., Size1      ! Make z-position of object
else                                ! equal to the larger of size1
> OBJ, POSITION, 0., 0., Size2      ! and size2
endif
```

Having **read and understood** this, the VR-Editor would **write** simply:

> OBJ, POSITION, 0.000000E+00, 0.000000E+00, 2.000000E+00

Once more, the Editor retains only the single-instance values;
but it **obliterates the declarations and condition** which led to them.

This can **very irritating!**

Some more history; three features needing protection

Relational input

PHOENICS User Meetings PARIS, 2008

1. In 1998 the **PLANT** feature was introduced into PHOENICS. This allowed **formulae** to be placed in the Q1 file, which after interpretation by the satellite, caused corresponding **Fortran coding** to be **created**, **compiled** and **linked** to the solver module.

2. Then in 2001 the **In-Form** feature was introduced. Its purpose and effect were the same, namely to allow users to **extend the simulation capabilities** of PHOENICS; but it did so **without requiring Fortran** coding to be created, compiled or linked into a new executable.

Both PLANT and In-Form statements had to be **protected** from the obliterating tendencies of the VR-Editor, by **'SAVE'** markers placed before and after them; these warned the Editor **to save the statements and place them properly** in the Q1 file which it was writing.

3. In 2007 it was recognised that a similar device could be used to protect those **advanced-PIL statements** (declarations, IF-statements, relationships, etc) which the Editor should not be allowed to obliterate.

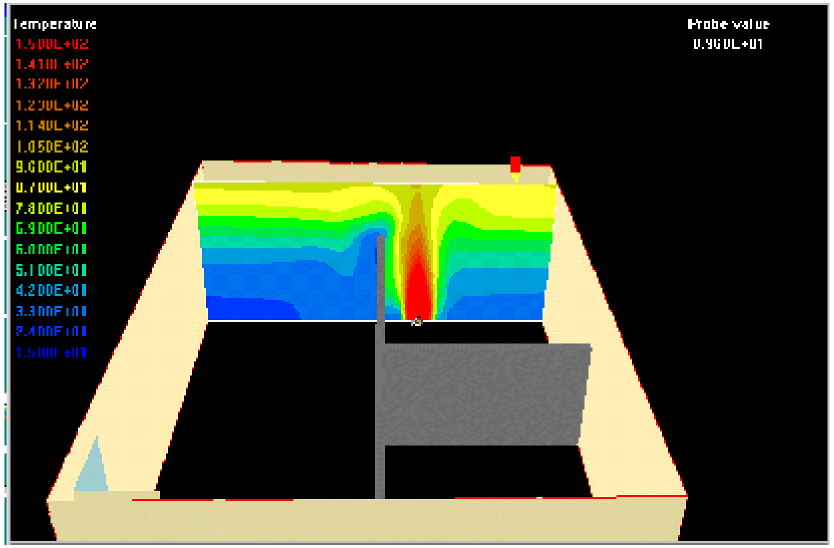
Thus came into existence the **'protected mode'** of satellite operation, the operation of which will now be illustrated.

A protected-mode example FLAIR-library case, I201

Relational input

PHOENICS User Meetings PARIS, 2008

The image on the right shows instantaneous **temperature distributions** calculated on the assumption that a fire is burning on the floor of a partitioned room.



The q1 file has been in the PHOENICS/FLAIR input-file library for many years as i201.

The 2008 version of this file will be used as an example of how the use of the **protected mode** of Satellite operation enables **relationships** to be expressed and preserved in Q1 files.

In effect, **all the features of advanced PIL** have now become available to those users of the VR-Editor who are willing also to **use its in-built text-editor**.

Differences between old and new i201.htm

Relational input

PHOENICS User Meetings
PARIS, 2008

Comparison of the old and new Q1s reveals that the latter has additional features, of which a few will now be described.

• The **new** file **declares logical variables** ‘zup’ and ’fourwall’ and **sets** them thus between **SAVE1BEGIN** and **SAVE1END markers:**

```
  SAVE1BEGIN           ! Marks start of section to be protected

  Group 1. Run Title
boolean(zup,fourwall)    ! declarations
zup=f                    ! settings
fourwall=t
TEXT( Room air flows; I201;  zup=:zup:

  Echo InForm settings for Group  1
  Group 1. Run Title
  SAVE1END              ! Marks end of section to be protected
```

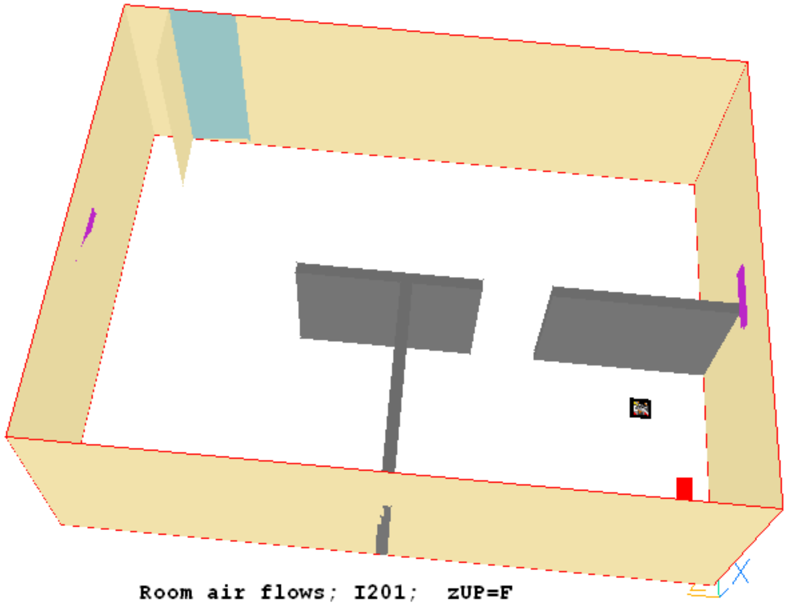
It suffices to explain only **‘zup’** . This stands for **z-direction is ‘up’** and has been introduced because the original file, contrary to current convention, used **x** as **‘up’**.

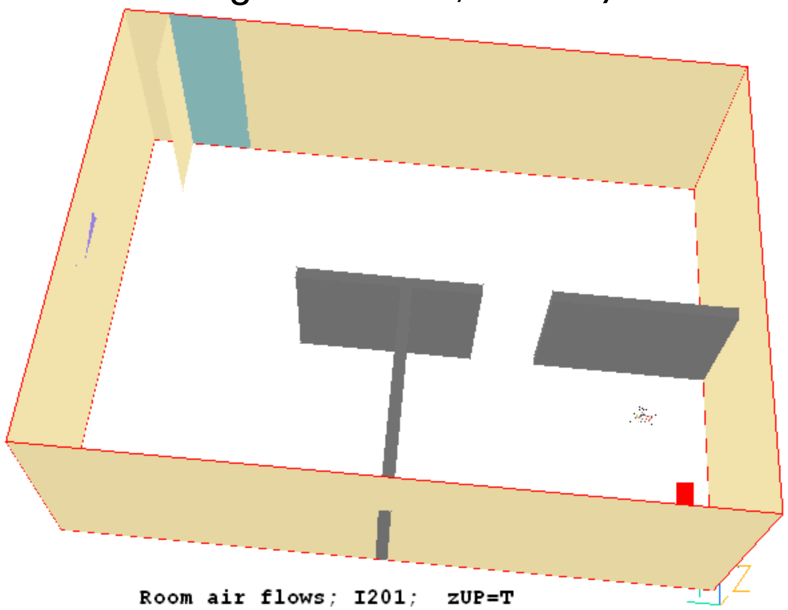
Use of the logical variable zup to change the ‘up’ direction

Relational input

PHOENICS User Meetings
PARIS, 2008

On the right is the first VR-Editor view when **zup=f**, its default value. But when, during the VR-Editor session, the Q1 file is hand-edited and **zup=t** is set, saving and loading the working files leads, **below,** to …





what looks like the same picture, but, closely examined, proves to have its **axes differently lettered**.

Advanced-PIL lines in the Q1 have made all the changes in response to the setting of a single variable, zup.

It is much harder to do this interactively!

Changing positions and sizes

Relational input

PHOENICS User Meetings
PARIS, 2008

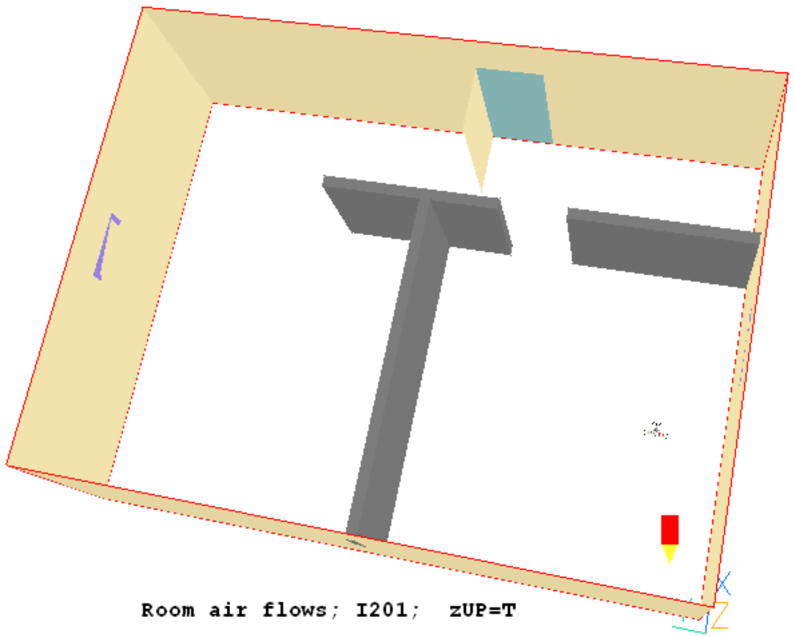
Here the door and partitions have moved.

This was effected by opening the q1 for editing **while still in VR-Editor mode**, and then finding and **changing three of the variables** which are declared there, namely:

'doorzpos', which governs the position of the door,

'doorhigh' which governs its height,

'prt1wide' which affects the width of the lowest (on the picture) partition.



Evidently, the wall aperture has **changed its position** and height to accord with the door; and all the partitions have **changed their sizes or positions** in order to preserve the relationships which are implied by the Q1.

Moreover, because they are **protected by 'SAVE'** markers, the relationships cannot be obliterated by the Editor, which dutifully writes precisely what it has read.

How the relationships are expressed in the Q1

Relational input

The **relationships** between the sizes and positions are expressed in the Q1 file by the lines printed on the right.

It is easy to understand their meanings, once it is remembered that they were written for the **non-conventional x-is-up z-is-along** co-ordinate system.

```
> OBJ,    NAME,       PART-1
xpos=0.0 ; ypos=0.0; zpos=prt1zpos
xsiz=prt1high   ; ysiz= prt1wide   ; zsiz=prt1thck

> OBJ,    NAME,       PART-2
xpos=0.0 ; ypos= prt1wide; zpos=0.0
xsiz=prt1high   ; ysiz= prt1thck   ; zsiz=prt2wide

> OBJ,    NAME,       PART-3
xpos=0.0 ; ypos=prt1wide; zpos= prt3zpos
xsiz=prt1high   ; ysiz=prt1thck    ; zsiz=prt2wide
```

How, it might be asked, was the **switch from the non-conventional system** to the conventional effected? The **following two lines**, appearing **after** the setting and **before** the use of the geometric attributes of **each** object, did all that was necessary:

```
 if(zUP) then
dummy=zpos; zpos=xpos; xpos=ypos; ypos=dummy
dummy=zsiz; zsiz=xsiz; xsiz=ysiz; ysiz=dummy
endif
```

Such are the tricks that a little knowledge of advanced PIL allows one to play.

Introducing new logic

Relational input

Suppose that it is desired, temporarily, to **remove the partitions** and/or the **fire** from the scene. This can be done very simply *via* the built-in editor during a VR-session, as follows, **namely by:**

1. in imitation of what has been done for ‘zup’ and ‘fourwall’, **declaring new boolean variables**: ‘nopart’ and ‘nofire’;
2. setting them = t or = f, as desired;
3. on the line above those defining partition-object attributes, **inserting** the lines;

```
if(nopart) then
goto nopart
endif
```

4. on the line below the attribute-defining lines **inserting:**

```
label nopart
```

5. **making** the corresponding **insertions** above and below the fire-object lines.

It will then be found that, **when the Editor is run**, the partitions and the fire are present or absent according to the settings of the respective variables.

This is another example of how the protected mode of operation allows useful variables to be **declared and used**, without, as hitherto, being obliterated.

Introducing interactivity

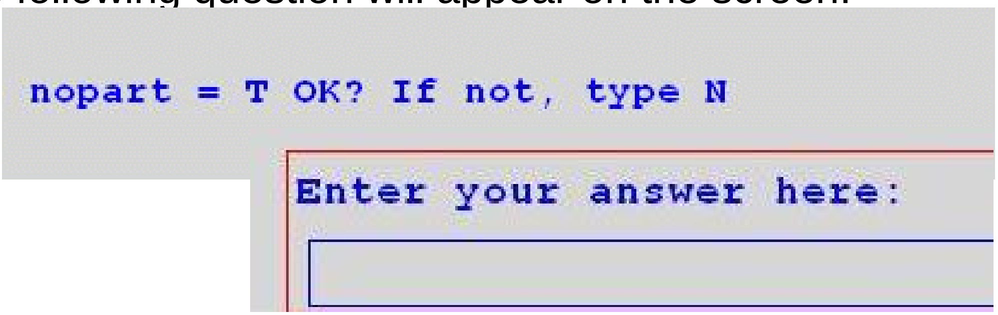
Relational input

PHOENICS User Meetings
PARIS, 2008

Advanced PIL allows **interactive modification** of settings. Thus, if the following lines are typed into the Q1:

```
mesg(nopart = :nopart: OK? If not, type N
readvdu(ans,char,Y)
if(:ans:.eq.N.or.:ans:.eq.n) then
nopart=f
endif
nopart
```

the following question will appear on the screen:

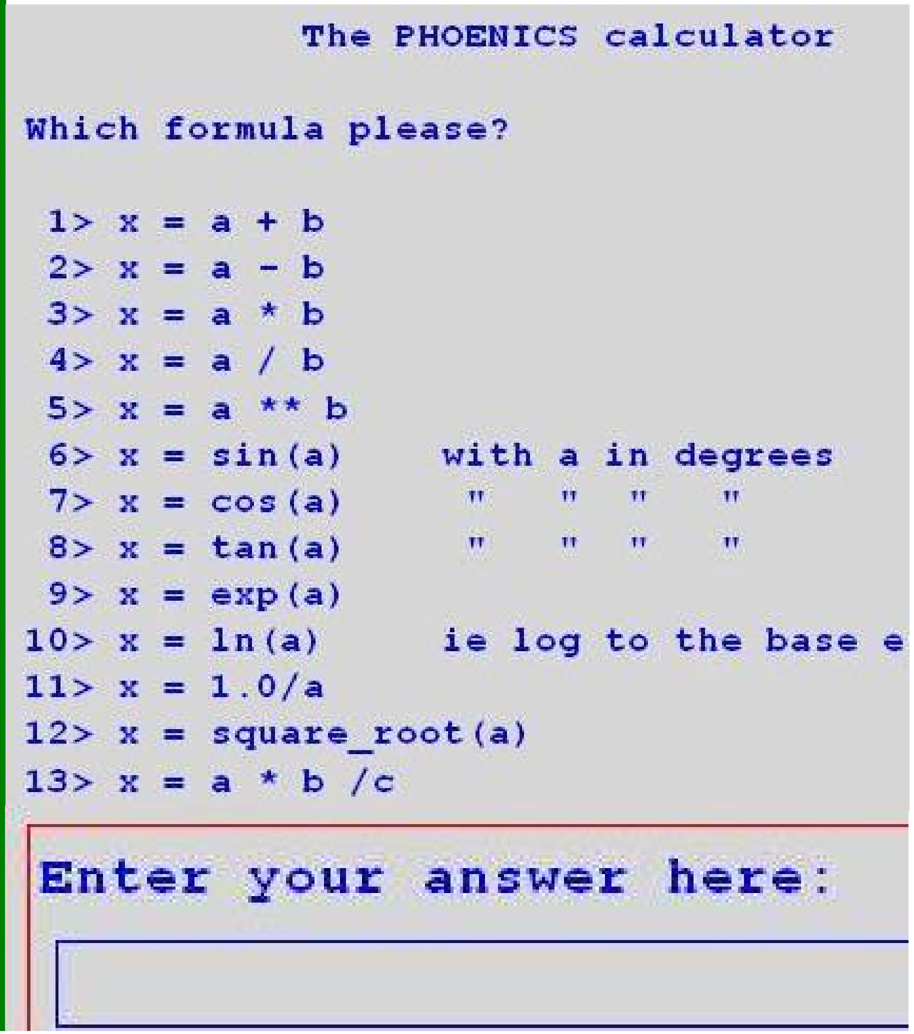


Typing N (or n) will then set nopart=F; then no partitions will be present to obstruct the flow in the room.

Introducing interactivity; the satellite as a calculator

Relational input

Loading core library case 011 into the PHOENICS satellite leads to the following:


```
            The PHOENICS calculator

Which formula please?

 1> x = a + b
 2> x = a - b
 3> x = a * b
 4> x = a / b
 5> x = a ** b
 6> x = sin(a)     with a in degrees
 7> x = cos(a)      "    "  "    "
 8> x = tan(a)      "    "  "    "
 9> x = exp(a)
10> x = ln(a)      ie log to the base e
11> x = 1.0/a
12> x = square_root(a)
13> x = a * b /c

Enter your answer here:
```

Evidently PHOENICS is offering to perform the role of a **calculator**; and it suggests some mathematical operations which its user might like to perform.

If some other operation is preferred, the user can **edit the file** 011.htm appropriately, so as to provide the additional formula.

Having typed the reference-number of the formula into the enter-your-answer box, the user is asked to **supply the values** of the constants a, b and c which are of interest.

Thereafter the **required result** appears instantly on the screen.

Advanced PIL is **worth learning!**

Introducing a new object

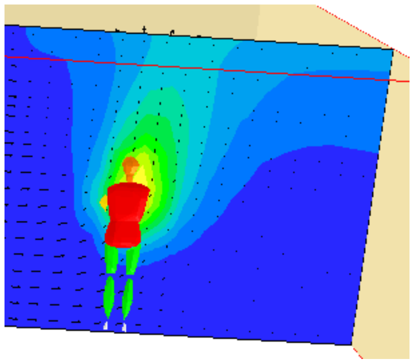
Relational input

PHOENICS User Meetings PARIS, 2008

New objects can be introduced interactively, as is well known. However, they can **also** be **introduced by hand-editing**.

Thus a user might have noticed the library case i200 contains a **standing man**, and wish to have one in i201 also. Then he or she could simply **copy the lines** from the relevant q1, perhaps modifying them slightly by use of xpos, etc.

```
xpos=1.500000E+00; ypos=2.000000E+00;zpos=
0.000000E+00
xsiz=3.000000E-01; ysiz=6.000000E-01;zsiz=
1.760000E+00
> OBJ,   NAME,        MAN
> OBJ,   POSITION,    :xpos:, :ypos: ,:zpos:
> OBJ,   SIZE,        :xsiz:, :ysiz:, :zsiz:
> OBJ,   GEOMETRY,    standing
> OBJ,   ROTATION24,      5
> OBJ,   TYPE,        PERSON
> OBJ,   POSTURE,     STANDING
> OBJ,   FACING,      +X
> OBJ,   WIDTH,       :ysiz:
> OBJ,   DEPTH,       :xsiz:
> OBJ,   HEIGHT,      :zsiz:
> OBJ,   SOURCE-FORM, Total-heat
> OBJ,   HEAT,        8.000000E+01
```



Then, if the partitions and fire have been removed and the solver activated, the picture on the left will appear in the corner of the room.

As the lines above dictate and the picture confirms, **the man is a source of heat**.

Relational input

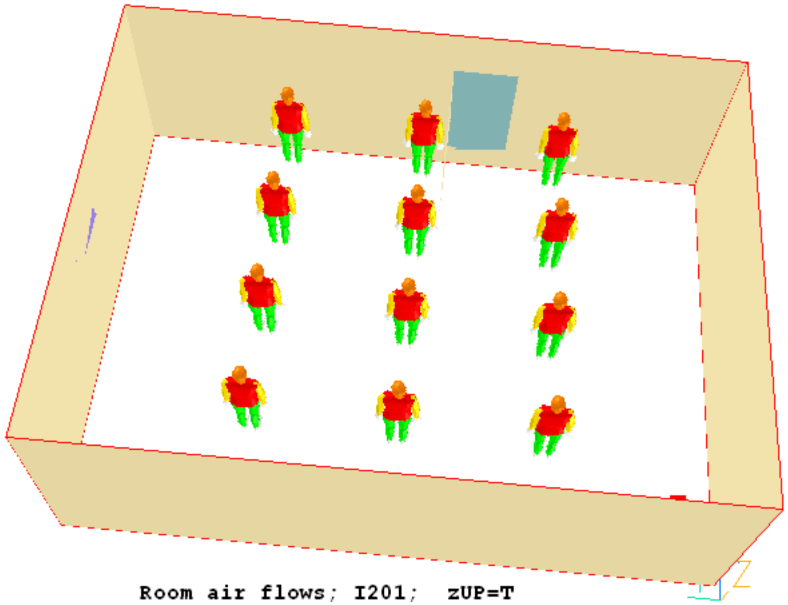
Introducing an array of objects

PHOENICS User Meetings
PARIS, 2008

If **one** man can be introduced, why not **many**? The **do loop** feature of advanced PIL makes this easy, as shown below:

```
do ixx=1,nmanx
do iyy=1,nmany
 xpos=1.500000E+00; ypos=2.000000E+00;zpos=
0.000000E+00
xsiz=3.000000E-01; ysiz=6.000000E-01;zsiz=
1.760000E+00
xpos=1.5*:ixx:;ypos=2.0*:iyy:; zpos=0.0
> OBJ,   NAME,       MAN:ixx::iyy:
> OBJ,   POSITION,    :xpos:, :ypos: ,:zpos:
> OBJ,   SIZE,        :xsiz:, :ysiz:, :zsiz:


> OBJ,   WIDTH,       :ysiz:
> OBJ,   DEPTH,       :xsiz:
> OBJ,   HEIGHT,      :zsiz:
> OBJ,   SOURCE-FORM,  Total-heat
> OBJ,   HEAT,        8.000000E+01
enddo
enddo
```



The picture above shows what results when the VR-Editor is activated. One can change the numbers of rows and columns by declaring and setting the variables: 'nmanx' and 'nmany'.

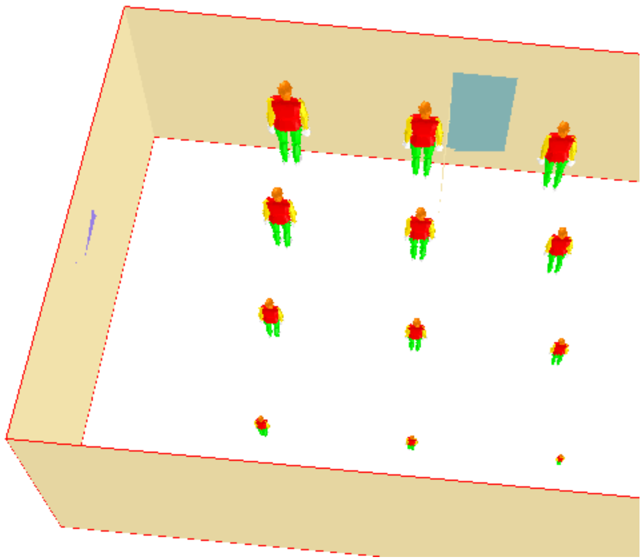
Changing their sizes

Relational input

PHOENICS User Meetings PARIS, 2008

The following further lines placed in the protected Q1:

```
real(shrink,factor)
factor=1/(nmanx*nmany)
shrink=factor
do ixx=1,nmanx
do iyy=1,nmany
factor=factor+shrink
xpos=1.5E+00; ypos=2.E+00;zpos=
0.0E+00
xsiz=3.E-01*factor; ysiz=6.E-01*factor;
zsiz= 1.76*factor
xpos=1.5*:ixx:;ypos=2.0*:iyy:; zpos=0.0
```



… will cause the sizes of the men to vary as shown above.

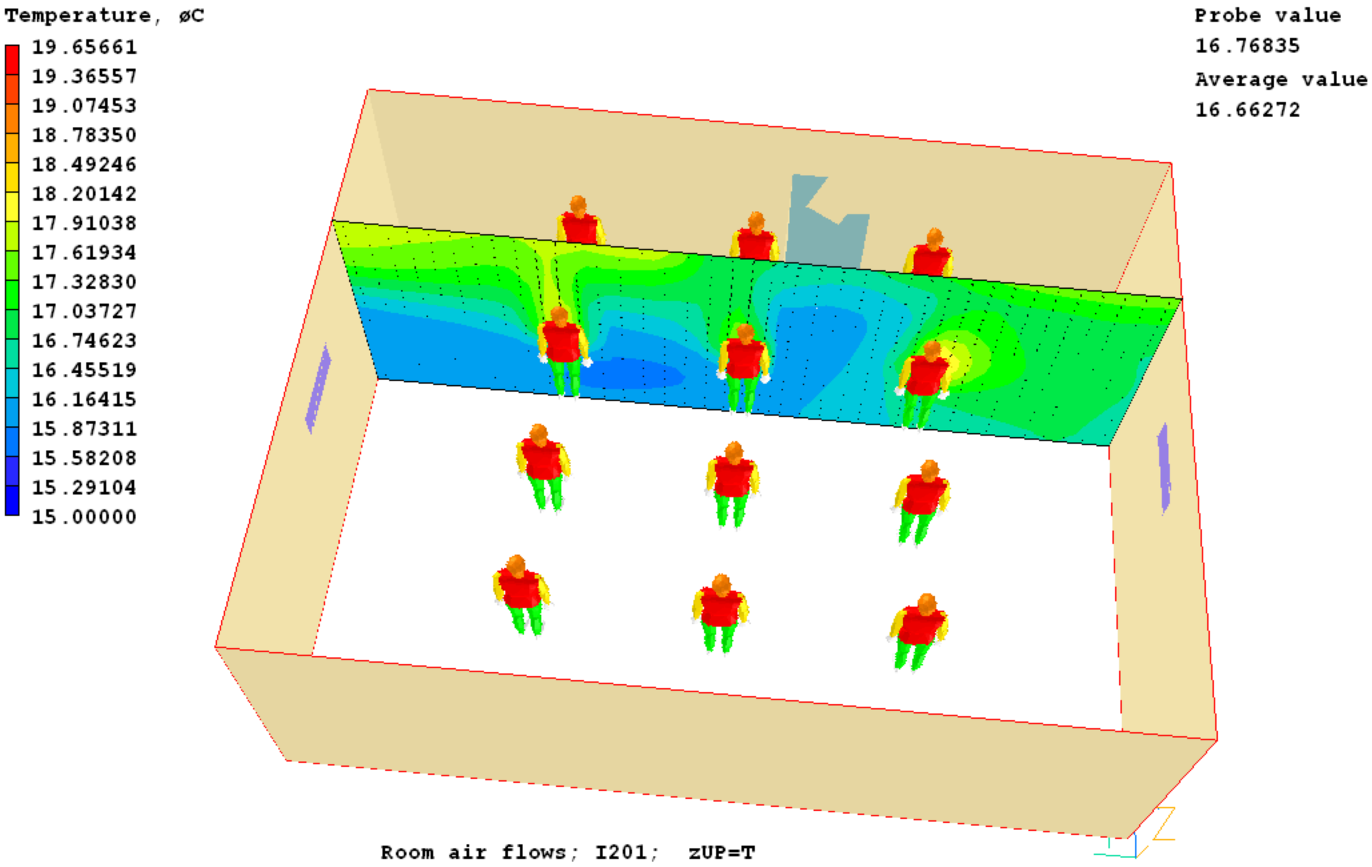
Of course, innumerable formulae for changing the sizes and positions could be devised; and the Editor will not obliterate them because they are 'SAVEd'.

Results (for many men)

Relational input

PHOENICS User Meetings PARIS, 2008

The results are quickly obtained by running the PHOENICS solver, and then the VR-Viewer; and they are as expected. See below, (for the **equally-sized** men). Warm air rises above each of them,



Summarising remarks about the use of protected-mode Q1s

Relational input

PHOENICS User Meetings
PARIS, 2008

• Protected-mode Q1s are **easier to read and to edit** than those created by the VR-Editor, because they contain more **understandable words** and **fewer** hard-to-comprehend **numbers**.

• If these words are the names of declared PIL variables, they can express **relationships** between the positions and sizes of individual objects.

• Moreover, **much more complex relationships** can be expressed than have been exemplified so far; and they can also contain non-geometric variables, such as **sources**, **initial values**, **material properties and time**.

• When PHOENICS users recognise what freedom the **'protected mode'** affords them, they will finally cease to feel **forced always to work interactively**.

•They will cease to be the **'prisoners of the mouse'**, **as** illustrated on the right**.**

• How to use the Advanced PHOENICS Input Language is explained in the PHOENICS Encyclopaedia.

But that's not all; there's PRELUDE!

Relational input

Why we need more:

1. Although the 'protected mode' does allow Advanced PIL to be exploited, **that language has some limitations.**

For example, although it **does** allow **one- or more-dimensional arrays** to be employed, their arguments **must always be integers**.

So it does **not** understand such constructs as:

xpos(door),

where **door** is an object name.

2.The VR-Editor does not itself allow the typing of expressions into its dialogue Boxes; nor does it provide any error-checking when the built-in text editor is used.

The answer? PRELUDE, the pre-pre processor, and its Gateways.

What PRELUDE provides

Relational input

PHOENICS User Meetings
PARIS, 2008

PRELUDE provides both **more** and **less** than the VR-Editor & Viewer.

The **more** includes:

- It can use **object names** as the arguments of its functions.
- **Expressions** can be typed into its dialogue boxes.
- The expressions can be of **unlimited complexity**.
- It provides **error-checking** and **'undo'** capabilities.
- It has a more flexible **position/size/rotation** language.
- It can handle many more **CAD formats**.
- It can launch **multiple runs** with systematic data-input variations.
- It can create parameterised objects by accessing Shapemaker.
- It stores its output in **multiple-instance Q3 files** instead of **single-instance Q1s**.

The **less** includes:

- It has still only limited results-display capability, so **uses the Viewer**.
- It **(deliberately)** offers users the **restricted choice** of data-input possibilities which is appropriate to the **'Gateway'** in question.

Gateways are the modern equivalent of **'Special-Purpose Programs'**.

PRELUDE and its Gateways

Relational input

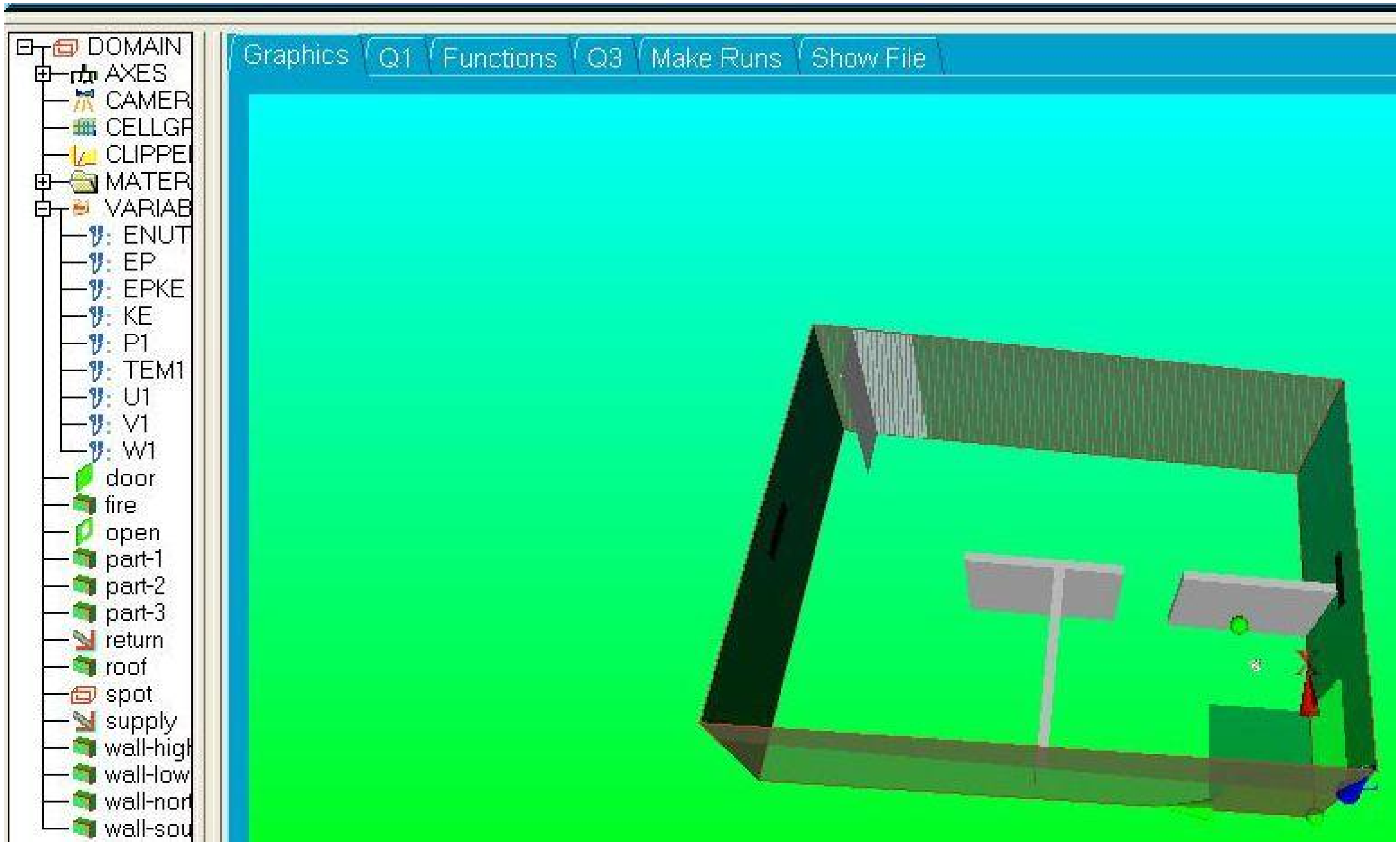
When PRELUDE is launched, it asks what is to be loaded; and it offers certain **'Gateways'**. These are quick-access routes to the particular features of PHOENICS which are likely to be useful to narrow-interest users.

PHOENICS-FLAIR users are likely to want to use the **HVAC** Gateway; but the others shown as available here are: **'Beginner'**, for those who want to learn; **'VWT',** for those who wish to use the **'Virtual Wind Tunnel'**; and **'HEATEX'**, for those who are concerned with heat exchangers.

PRELUDE and its Gateways; the ‘roomfire’ scenario

Relational input

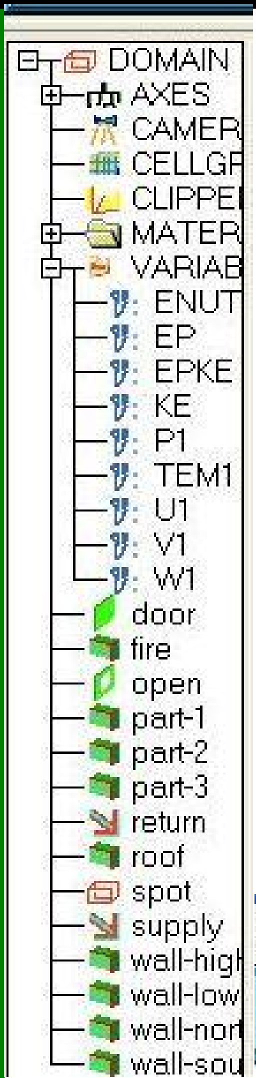
If HVAC is selected, another menu will appear. Then selection of the item called ‘**roomfire**’ will load a scenario which has been designed to resemble closely that of **library case I201** which has been discussed above.



PRELUDE and its Gateways; the 'room-fire' scenario

Relational input

PHOENICS User Meetings
PARIS, 2008

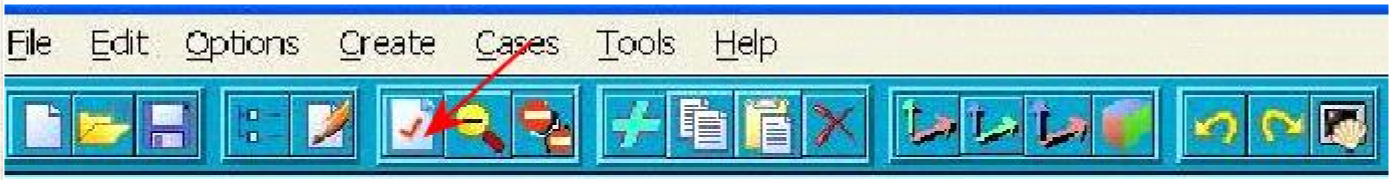


On the left of the image of the scenario, PRELUDE displays the so-called '**object tree**'.
At its top are PRELUDE-specific objects, such as are explained in the **tutorial** supplied with the Beginners Gateway, begin1.htm

Then follow items which are **familiar to PHOENICS users**; specifically the names of the solved-for and whole-field-stored **variables** are listed, each being treated as a '**virtual** object' having definable attributes.

Below them will be seen the names of the **substantial** objects which constitute the **scenario:** fire, door, open(ing) and the partitions, walls *etc,* which were encountered in library case i201.

Their **attributes can be revealed** by clicking on the object name, so as to select it, and then on the red-tick icon in the **tool-bar** shown below.



PRELUDE and its Gateways; attributes of objects

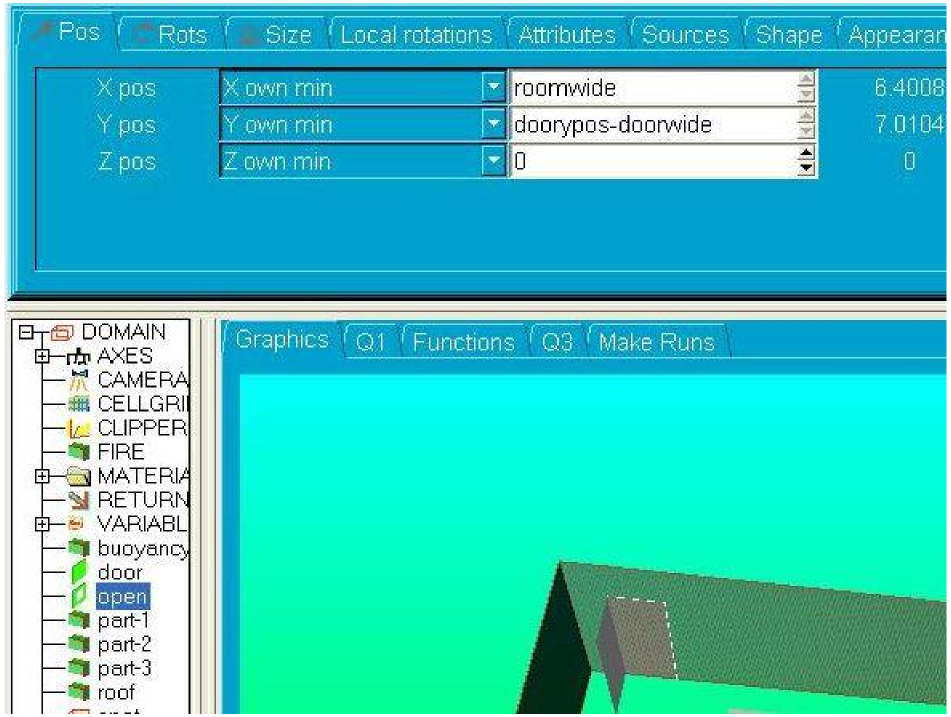
Relational input

PHOENICS User Meetings
PARIS, 2008

Here for example are the **attributes** of the more-conventional **object** called **‘open’,** the aperture in the wall which can be closed by the door.

Its attributes are revealed in the white boxes by clicking on its name in the tree and then on the **red tick** of the top-menu bar.

These attributes are understandable **expressions**; thus its y-position is given as **‘doorypos-doorwide’**.



Therefore, if the door is moved, the opening will move with it, just as occurred when the scenario was described by a ‘protected Q1’, earlier in this presentation.

Moreover PRELUDE can handle **more-flexibly-formulated** expressions. Thus: **‘ypos(door)-ysize(door)’** would have the **same significance**, and obviates PIL’s need to declare the non-standard variables: **‘doorypos’** and **‘roomwide’**.

PRELUDE and its Gateways; attributes of objects (continued)

Relational input

PHOENICS User Meetings PARIS, 2008

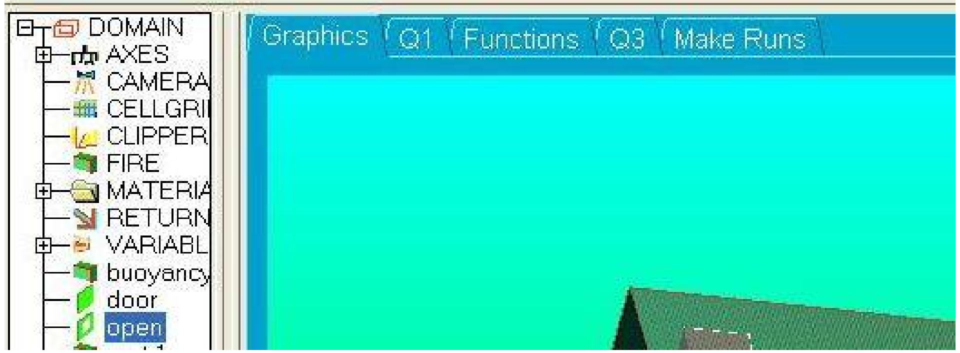
‘OPEN’ has of course **other attributes**, as this image shows. They are the same as would appear in a Q1 file.

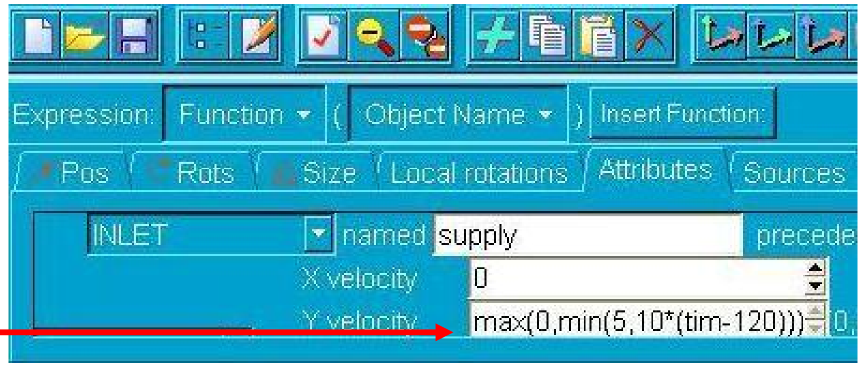
It has the standard FLAIR ‘**type**’, namely ‘**opening**’; and a **pressure coefficient** allowing air to enter or leave.

However, PRELUDE allows **more complex** entry and leaving relationships to be specified than the VR-Editor can envisage.

Suppose one wishes to make the inflow through the ‘supply’ port **at first** 0.0, **rising** to 5 m/s after 120 seconds, when the fire starts.

This can be achieved as shown here.

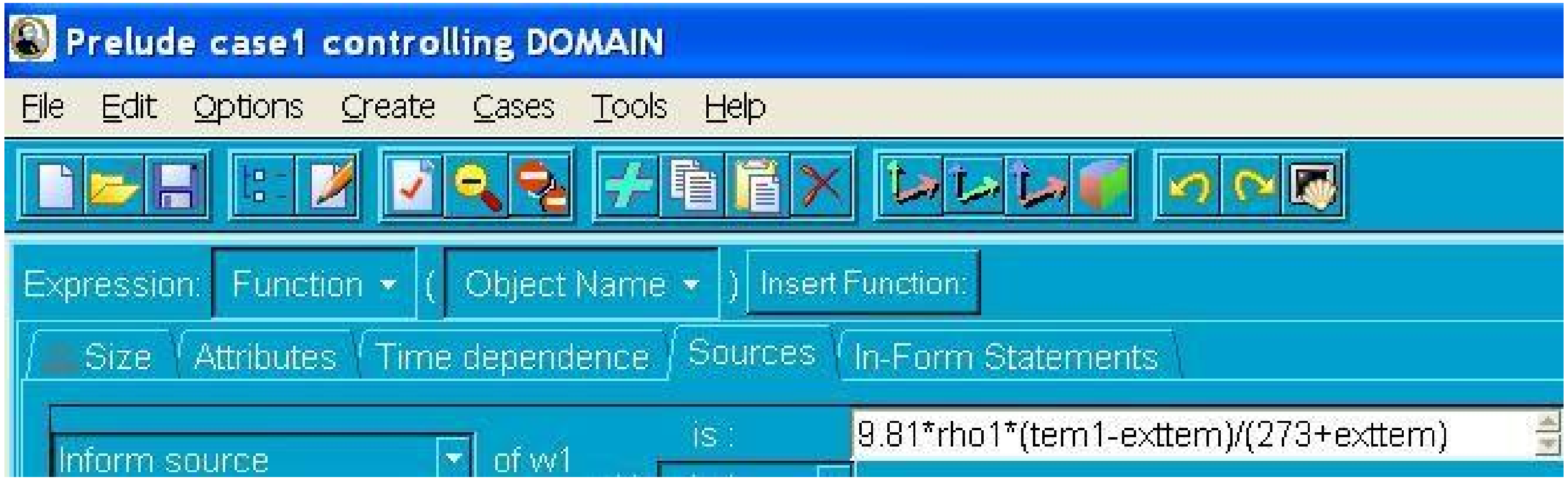




PRELUDE and its Gateways; how buoyancy is represented

The interaction of the force of **gravity** with the density variations caused by temperature changes can be introduced by way of a **buoyancy object**.

In the present example however, the practice of i201 is emulated by way of a **source of vertical-direction momentum**, *i.e.* W1, the z-direction velocity.
This is treated as an attribute of the **domain,** because gravity acts everywhere.



Here 9.81 is the **gravitational acceleration**, rho1 is the reference density, exttem is the **external temperature** (15 degrees Celsius) and tem1 is the **local temperature** of the gas.

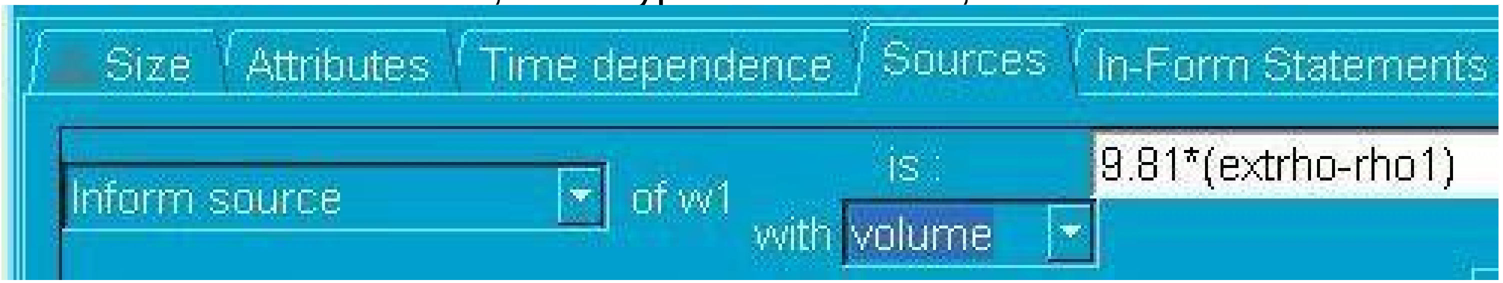
The formula can be recognised as expressing the '**Boussinesq approximation**'.

PRELUDE and its Gateways.
Modifying the **buoyancy object**

Relational input

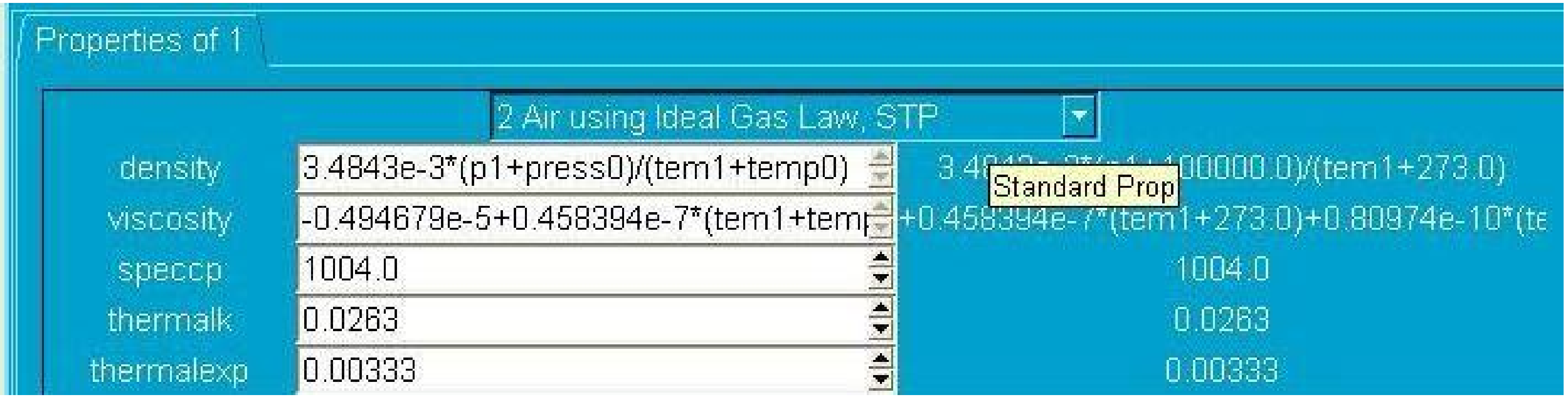
PHOENICS User Meetings
PARIS, 2008

The Boussinesq formula is accurate only **when the temperature variations are small** compared with the absolute temperature. For flames, a **more appropriate formula** for the w1-source, to be typed into the box, is that shown below.



Extrho is the external density and rho1 is the **local** density, which of course must be calculated appropriately.

If the ‘hot-air’ combustion model of library case i201 is retained, the appropriate formula is the **Ideal-Gas Law**, summoned in Prelude by a few mouse clicks, thus



PRELUDE and its Gateways: attributes of the **fire object**

Relational input

The **fire object** is represented in the same manner as in library case i201, namely as a fixed-flux heat source of magnitude ‘fireflux’ which has been set as 70 kilowatts.

However, some specialists believe that the true heat input of a fire can **never** be fixed; for it must fall to zero when the adiabatic combustion temperature (*e.g.* 2000 degrees) is reached, signifying that **all the oxygen has been consumed**.

This is easily expressed by typing not **fireflux** but **fireflux*(1-tem1/2000)**. PRELUDE allows this; and the PHOENICS solver will act accordingly. The following image shows what will appear on the screen.

PRELUDE and its Gateways;
the **need for new variables**

Relational input

The just-described device for **limiting the attainable temperature** is certainly an advance over the fixed-heat-flux practice.

However to represent combustion processes **more realistically**, it is necessary to calculate the state of the gas mixture in more detail; and this means **solving for more variables**.

The variables which are solved **by default** in the roomfire Gateway have already been seen. Those **solved** are: P1, TEM1, U1, V1, W1, KE and EP; while those **auxiliary variables** which are only **stored** are: ENUT and EPKE.

A more complete representation of combustion conventionally needs also:
the **FUEL** mass fraction, a measure of the fuel/air ratio **MIXF**, and the enthalpy **H1**.

Which are to be **solved** and which only **stored as auxiliary variables** depends on further decisions as to whether:

1. the **'mixed-is-burned'** presumption is true or false; and
2. the flow is or is not presumed to be without **heat loss** to the solid surroundings.

PRELUDE allows these decisions and their consequences to be expressed in a **simple manner**.

PRELUDE and its Gateways; adding new variables

Relational input

PHOENICS User Meeting
PARIS, 2008

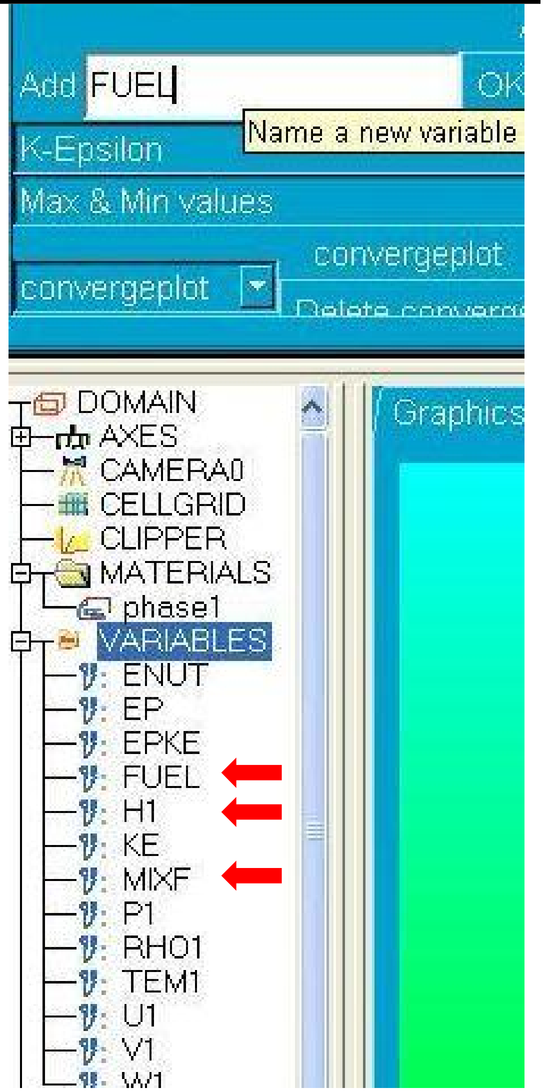
Adding new variables is easy with PRELUDE.

If the object 'variables' is selected, by clicking on its name in the tree, and then the red-tick attributes icon is clicked, an **'add a variable'** opportunity is provided.

Typing into the white box H1, MIXF and FUEL, and clicking OK after each, increases the contents of the object tree as shown on the right:
the **desired variables have been added** (and RHO1 also, so that density can vary).

The next question to consider is: which should be **solved-for** variables and which **stored-only**?

Whatever the answer, PRELUDE provides an easy means of expressing it, as the next slide shows.

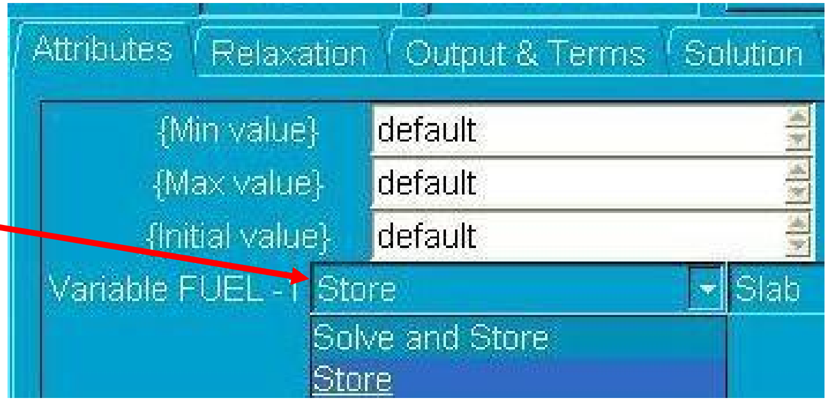


PRELUDE and its Gateways; solved-for or stored-only variables

Relational input

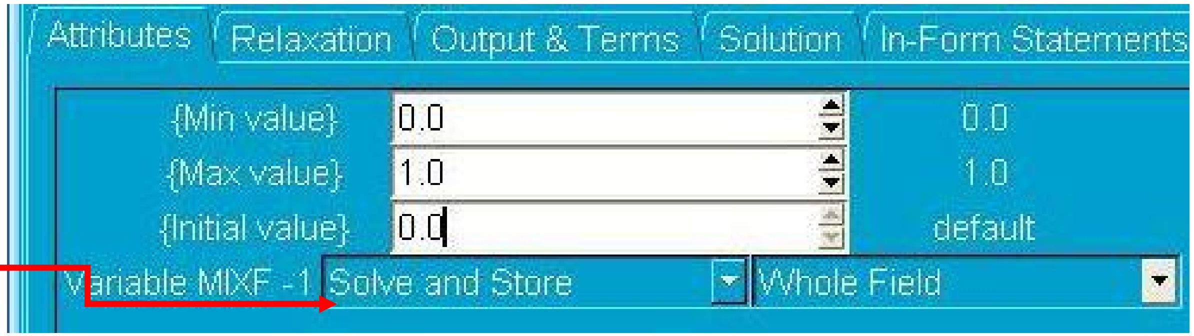
PHOENICS User Meetings
PARIS, 2008

On the right is the menu which is offered when **FUEL** is selected and its **attributes** (red-tick box) are called for.



The option 'store' has been selected, because the simplest combustion model will be chosen first, embodying the **'mixed-is-burned'** presumption.

That model needs however that MIXF should be both stored and **solved**. That choice is shown on the right.



More choices are also shown, namely that the '**whole-field**' method of solution is to be chosen, that zero and unity shall be the minimum and maximum values which MIXF is allowed to attain, and that its initial value shall be zero.

Such settings, commonly set *via* the VR-Editor, **can** be set *via* PRELUDE menus; but there is no **need**; for Gateways are provided with **acceptable defaults**.

PRELUDE and its Gateways; the choice of combustion model

Relational input

The four combustion models which it is especially appropriate to introduce are:

1. Mixed-is-burned; **adiabatic**.
For this, one **solves** only for **MIXF**, and stores **FUEL, H1** and **TEM1** which can be deduced from it.

2. Reaction-rate-limited; adiabatic
For this, one solves for **MIXF** and **FUEL** and stores **H1** and **TEM1** which can be deduced from them.

3. Mixed is burned; non-adiabatic.

For this, one solves for **MIXF** and **TEM1**, and stores **FUEL** and **H1.**

FUEL can be deduced from **MIXF;** and so can H1, which must however now be interpreted as the enthalpy which would prevail **if the flow were adiabatic**.

From H1 one can deduce **TEM1_adiabatic** which it is also useful to store; then the **TEM1** for which one solves has the significance of the **actual temperature minus TEM1_adiabatic**.

4. Reaction-rate-limited; non-adiabatic.

This is like 3, but with **FUEL** also solved for and influencing **H1**.

Relational data-input to PHOENICS: interim remarks

Relational input

How PRELUDE facilitates the introduction of the various combustion models must be left to **another presentation**.

The **limited aims of this presentation** have been to explain and exemplify that:

- **the ability to enter relational data** is an indispensable requirement for a modern CFD code;
- this is **now provided**, to some extent, by the PHOENICS VR-Editor in **'protected mode'**, which permits the use of all the features (declarations, logic, screen-keyboard interaction, file-handling, *etc)* of the years-old **Advanced PHOENICS Input Language**;
- but **PRELUDE surpasses** that provision by permitting **more-complex relationships** and supplying **as much interactivity as is needed** for each particular 'Gateway'.

Nevertheless …

Relational data-input to PHOENICS: what can be done **without** PRELUDE

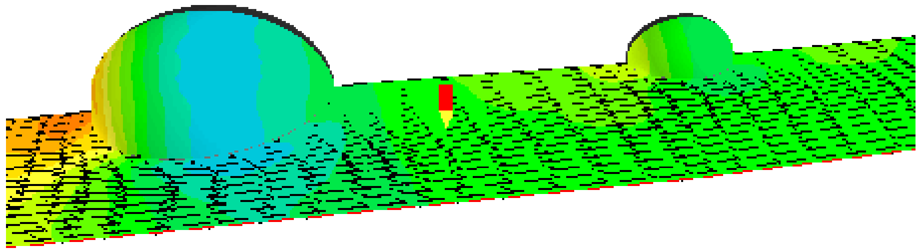
Relational input

PHOENICS User Meetings
PARIS, 2008

… lest attention to PRELUDE overshadow what can be done **without** it, a **hydrodynamic example** will be discussed.

This concerns flow past objects in a **wind tunnel**, and how its investigation is facilitated by the VR-Editor in **protected mode**.

Here is an example of what will be shown: two spheres, one behind the other.



2 spheres Re=40 quarter=T finegrid=T

This might be an exercise given to **students**, whose attention is to be **focussed** on just those aspects which their **professor** has been lecturing upon.

The focussing feature makes PHOENICS a useful teaching tool.

The Q1 file can be accessed by clicking here.

Like all library files, it can be loaded into the VR-Editor; then the users can make **any** desired change of input data.

But students, like most of us, require guidance: **helpful signposts;**

but **not too many** of them!

The PHOENICS Input Language allows **teachers** to provide these.

PHOENICS specialists in a company can do the same for their **design-department** colleagues who then, too, can **'do CFD'**.

The flow-past-spheres example:
Input File Library Case 807 (continued)

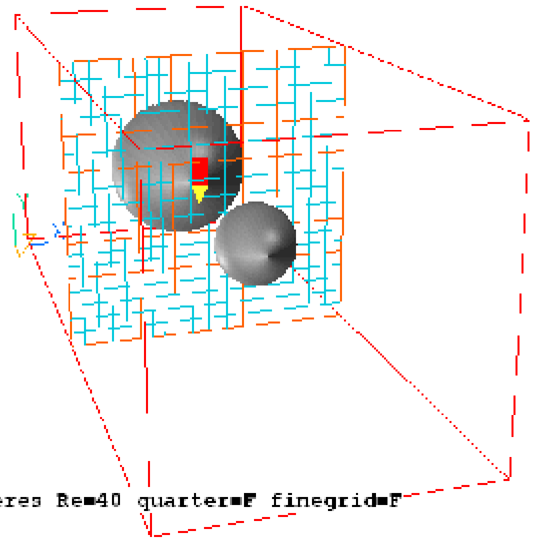
Relational input

In the case 807 Q1 is written: "*Provision is made for:*

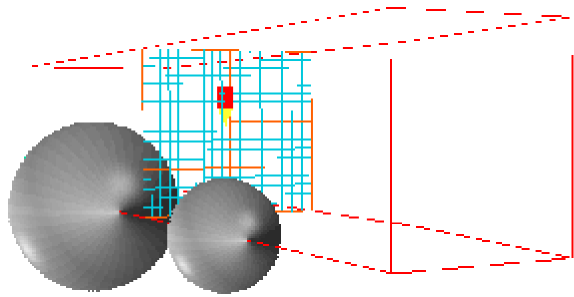
1. *Solving for only **one quarter** of the domain; this is allowed, by reason of symmetry, and desirable for economy and accuracy.*"

This means choosing between

this '**wholly-inside**' situation or this '**quarter-inside**' one:



2 spheres Re=40 quarter=F finegrid=F



2 spheres Re=40 quarter=T finegrid=F

PHOENICS allows both (and many more); but the **Q1 author** made **just these two** easily accessible.

PIL empowers!

The flow-past-spheres example: Input File Library Case 807 (continued)

Relational input

PHOENICS User Meetings PARIS, 2008

How did the Q1 author do it? By declaring and setting the variable: '**quarter**' (and **finegrid** and **reyno**) in the Q1 thus:

```
  SAVE25BEGIN
declarations and settings
boolean(quarter,finegrid)
real(reyno)
quarter = t ; finegrid= t ; reyno=40
```

Then, lower down in the Q1 are to be found:

```
…                    ! Set positions and sizes for quarter=f
If(quarter) then
….                   ! Modify positions and sizes
endif
```

Reminder: in PIL, **t** means true, **f** means false.

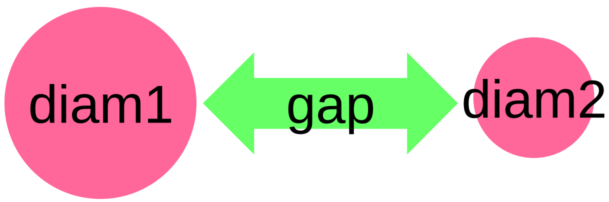
The flow-past-spheres example: Setting positions and sizes

In **unprotected** mode, the editor accepts sizes and positions for each object in a single scenario and records them as **numbers**. That's OK.

In **protected** mode, users can create a **range** of scenarios and can record sizes and positions as **relationships;** which is **much better**.

More **freedom** demands more **thought**: *e.g.* which shall be the **key parameters**? Which the **derived** ones?

The case-807 author chose **diam1**, **diam2** and **gap** as keys, thus:



These can be used as **parameters** in a systematic study of what influences the flow, the drag, the accuracy, *etc*.

The flow-past-spheres example:
Setting sizes and positions in the Q1

Relational input

Here are **some of the lines** which the 807-author wrote in the Q1:

declarations

```
real(diam1,diam2,gap)
real(xpos1,ypos1,zpos1,xsiz1,ysiz1,zsiz1,dist)
real(xpos2,ypos2,zpos2,xsiz2,ysiz2,zsiz2)
real(xposg1,yposg1,zposg1,xsizg1,ysizg1,zsizg1)
real(xposg2,yposg2,zposg2,xsizg2,ysizg2,zsizg2)
```

settings

```
diam1=2.0; diam2=1.0; gap=2.44
xulast=2.0*diam1; yvlast= 2.0*diam1; zwlast= 5.0*diam1
xpos1=diam1*0.5; ypos1=diam1*0.5; zpos1=1.11*diam1
xsiz1=diam1; ysiz1=diam1; zsiz1=diam1 etc
```

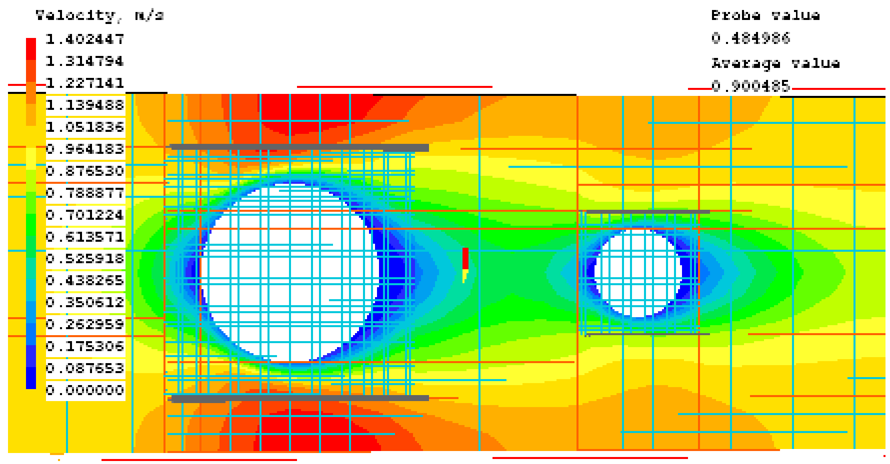
Tedious and mechanical! but written **once only.**

Thereafter **innumerable** runs result from changing one or more of these numbers. **Systematic studies can begin.**

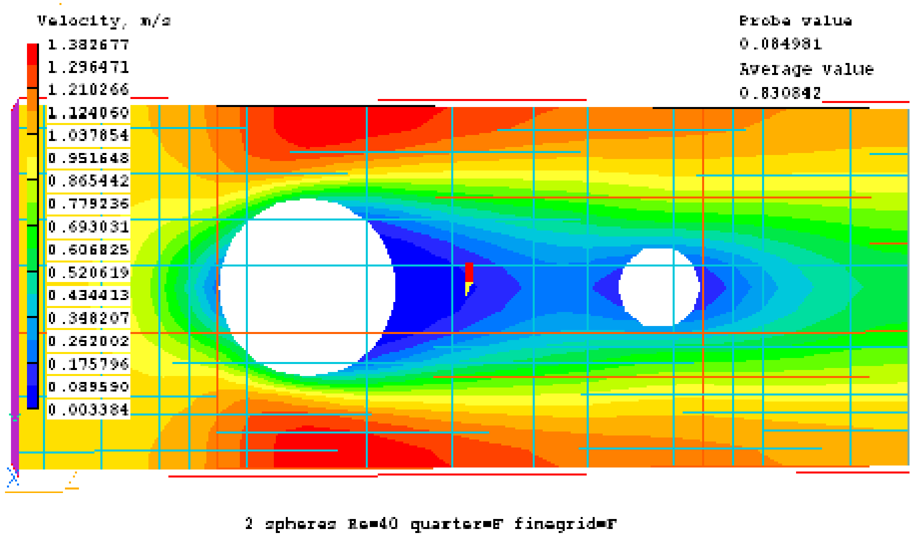
The flow-past-spheres example: A few results: the effect of ‘finegrid=t’

Relational input

It is interesting to compare the solutions with and without the fine grids. First for the full domain.



The solution without the fine grid is shown here.



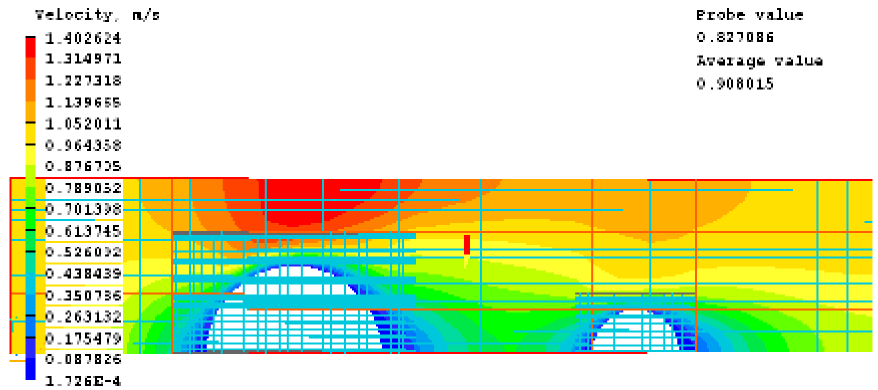
Although qualitatively similar, the differences show that the finer grid was indeed needed.

The flow-past-spheres example:
A few results: the effect of ‘quarter=t’

Relational input

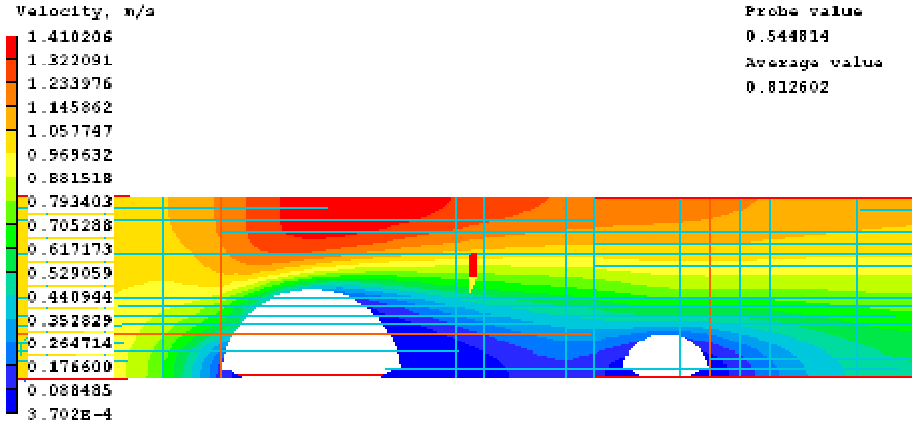
PHOENICS User Meetings
PARIS, 2008

And now the same comparison for the quarter domain.



The solution without the fine grid is shown here.

Although the maximum velocities are closer, the contours show at least a display flaw at the base.



2 spheres Re=40 quarter=T finegrid=F

The flow-past-spheres example: A closer look at the solution

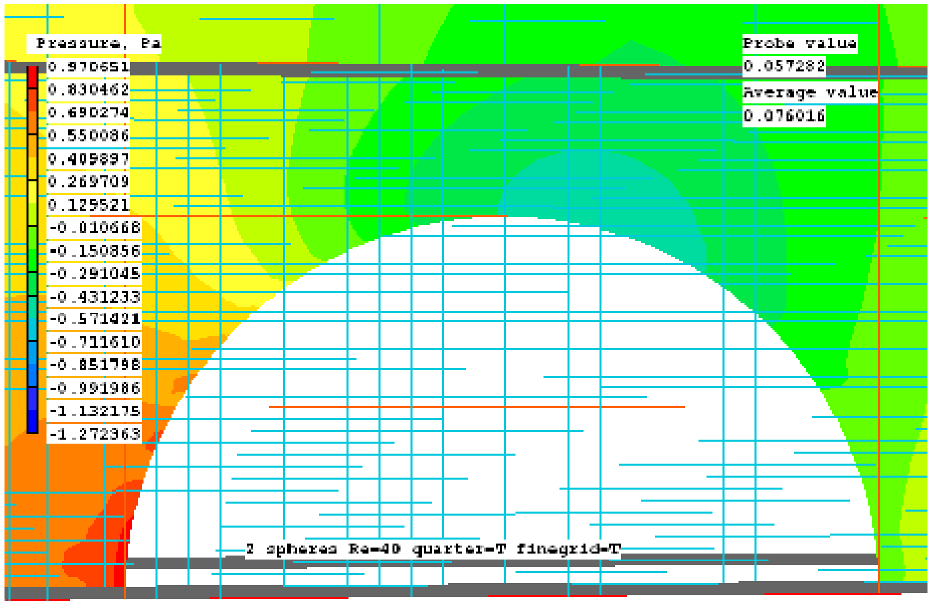
Relational input

PHOENICS User Meetings PARIS, 2008

In all these computations, the PHOENICS variable PARSOL = t. This means that the mass- and momentum-conservation equations for the ‘**cut cells**’ at the sphere surface were given special treatment.

The **smoothness of contours** there needs to be examined.

The contours of **pressure** are shown here. Their **smoothness** is **very good** despite the fact that the grid cells are not extremely small.



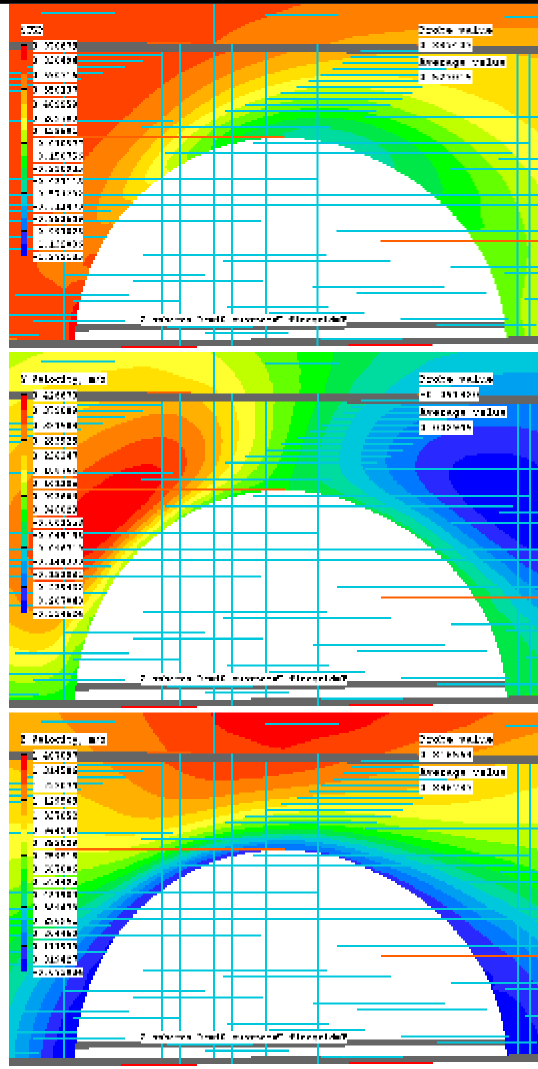
The flow-past-spheres example: A closer look at the solution

Relational input

The same is true for any of the computed variables. Here are shown contours for :

- stagnation pressure,
- y-direction velocity and
- x-direction velocity.

All are as smooth as can reasonably be desired.



PARSOL, because it completely **obviates the tiresome grid-generation** problems which beset other codes, is regarded by users of PHOENICS as one of its best features

The flow-past-spheres example: remarks about the parametric study

Relational input

This simple study would have been difficult **without** use of the parameterised Q1, now permitted by the **protected mode.**

Users' labour can be still further reduced by using the PHOENICS **'multi-run'** capability (*i.e.* RUN(1, **any number**)), by introducing into the Q1 such sequences as:

```
if(irun.eq.1) then
quarter = t
finegrid=f
endif
if(irun.eq.2) then
quarter = t
finegrid=t
Endif

etcetera
```

Reynolds number, diameter ratio, grid-refinement factors, iteration numbers and other influences can be varied **run-by-run.**

In this way, PHOENICS can be set to work for a **complete weekend**, and to present comprehensive results on Monday morning.

Interactive use of the VR Editor is OK for making **single runs**, but…

research requires parameterised Q1s.

Relational data-input to PHOENICS; concluding remarks

Relational input

PHOENICS User Meetings
PARIS, 2008

In 2008, significant advances have been made in the ability of PHOENICS to accept **relations** rather than single settings as input.

Two developments have effected this:

1. The **protected mode** of satellite operation, and
2. The pre-pre-processor PRELUDE.

Their advantage is similar in nature to that of the Excel spread-sheet over the hand-calculator.

Teachers can use the facility to focus the attention of their students.

Parameterised Q1s can be used by those without time or patience to learn to interact with the VR-Editor.

Research-minded users of PHOENICS can now proceed faster.

The end

How to learn about PRELUDE and its Gateways

Relational input

PHOENICS User Meetings
PARIS, 2008

The top menu bar of PRELUDE contains a ‘help’ button. Clicking on it will evoke a drop-down menu, containing the names of the PRELUDE tutorials which are present on the machine which is being used, which will probably include:

- begin1, a long tutorial which explains all the main features of PRELUDE;
- vwt1, which explains how to use the Virtual-Wind-Tunnel; and
- oneroom, which concerns simulation of the flow of heat and air in a ventilated room.

Each tutorial is contained in an html file which users are invited to read by means of a browser in one window while PRELUDE is open in another window.

There is also a document regarding PRELUDE, its purpose and its capabilities, which can be viewed here.