# Creating Parameterized Q1 files by means of the PQ1Editor

By: Alexey Ginevsky, Elena Pankova and Brian Spalding

(last edited on 24.12.15)

## Contents

## 1. Background

### 1.1 History

From the beginning of PHOENICS in 1981, the file which its users employed for the input of data has been called Q1, written in the PHOENICS Input Language, PIL; and this language allows variables to be declared, to be given values, and to be used as parameters in algebraic and logical expressions. The so-called ‘parameterized Q1s’ (known as PQ1s) were thus input files for **classes** of flow simulation, of which **particular instances** could be selected by causing the Q1 to include files containing particular sets of values for its parameters.

Such files had until a few years ago to be created by a text editor; but in the year 2011 PHOENICS-Direct was introduced, as a graphical user interface which allowed its users to select the parameters by way of easy-to-use menus. The PQ1-plus-PHOENICS-Direct package became known as a Simulation Scenario, or more recently as an app.

Included within the package was an intermediary file called **scene.xml**. This expressed the parameter-setting content of the PQ1 in a manner that PHOENICS-Direct (P-D from now on) could understand; and the creator of the PQ1 had in the beginning to create that file too. To **remove** that necessity, and its proneness to error, the PQ1Editor was created. This program, the subject of the present document, facilitates the writing of the PQ1 in a strictly-formatted manner; and, when that task is complete, **automatically** creates an entirely compatible scene.xml. Indeed, if appropriately instructed, it can create also the complete file structure of an entirely new SimScene.

### 1.2 Purpose of the present document

The main purpose of the present document is to explain to the PQ1 creator how to use the PQ1Editor (Pq1ed, from now on) for the just-mentioned tasks.

It should be understood that PQ1ed may of course be used for editing files of **any** kind. When however it is informed that the file loaded into it is to be used as a PHOENICS Input file, *i.e.* as a Q1, whether parameterized or not, it displays characters on the screen in special ways. It uses its built-in knowledge to distinguish active PIL and In-Form statements from each other, and from settings and from comments, by the selective use of: emboldening, Roman or italic, and varied colour. For example, a PIL statement is printed in bold font when, because it starts in the first or second column, it will be accepted as an instruction by the Satellite**.** When moved two spaces to the right however, which renders it inactive, its emboldening is removed.

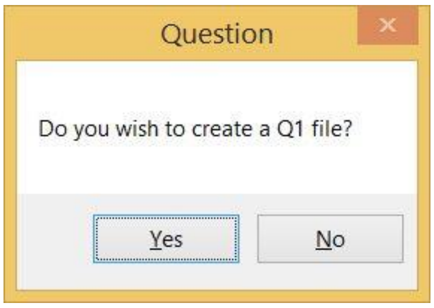
Such distinctions assist the user of PQ1ed to avoid mistakes.

## 2. Creation of P-D-compatible PQ1

### 2.1 *Ab initio*

**(a) Declaring the file as a Q1**

When Pq1ed is first opened the following message appears on the screen:


Question

Do you wish to create a Q1 file?

Clicking ‘yes’ causes Pq1ed to activate its above-mentioned Q1-related text-display features. Also, when the time comes to save the file, its name will be added to a list of Q1files which the editor maintains in \common\pq1ed\pq1ed.q1s. Had the answer ‘no’ been given, its name would later be added to the file pq1ed.noq instead

Pq1ed consults these files whenever a file is loaded into it, so that it need not ask users questions to which the answer is already known.

The user is then free to create the text of the file in whatever way he wishes; but Pq1ed is ready to minimise his labour, as will now be explained.

**(b) The use of pq1ed**’s **macro feature.**

A good way to start the creation of a PQ1 is to enter **pq** followed by <**Ctrl+j**>. This will bring to the screen the following extensive macro, *i,e,* prepared-in-advance text segment:

```
<html><head><title>q1.htm</title></head><body><pre><strong>
........................................................................
 PQ1 description
........................................................................
  Title:
Author
Start Date:
  Last editing date:
  Purpose:
Brief description:
  ************************************
 PQ1 Part 1 Declarations and settings
  ************************************
xml-group general
insert declaration and settings below

xml-group geometry
insert declaration and settings below
```

```
xml-group variables solved
insert declaration and settings below

xml-group material properties
insert declaration and settings below

xml-group model choices
insert declaration and settings below

xml-group initial condition
insert declaration and settings below

xml-group boundary conditions
insert declaration and settings below

xml-group output settings
insert declaration and settings below

xml-group computational grid
insert declaration and settings below

xml-group other numerical
insert declaration and settings below

xml-end
  ************************************
  PQ1 Part 2 Include Frommenu
  ************************************
INCL(frommenu.htm)

  ************************************
  PQ1 Part 3 Consequential settings
  ************************************
TALK=T;RUN(1,1)
GROUP 1. Run title and other preliminaries
TEXT(                              )
GROUP 2. Transience; time-step specification
steady= t or f
nregt=1
iregt=1;grdpwr(t,lstep,tlast,power)
GROUP 3. X-direction grid specification
nregx=1
iregx=1;grdpwr(x,nx,xulast,power)
GROUP 4. Y-direction grid specification
nregy=1
iregy=1;grdpwr(y,ny,yvlast,power)
GROUP 5. Z-direction grid specification
nregz=1
iregz=1;grdpwr(z,nz,zwlast,power)
GROUP 6. Body-fitted coordinates or grid distortion
GROUP 7. Variables stored, solved & named
GROUP 8. Terms (in differential equations) & devices
GROUP 9. Properties of the medium (or media)
GROUP 10. Inter-phase-transfer processes and properties
GROUP 11. Initialization of variable or porosity fields
GROUP 12. Patch-wise adjustment of terms (in differential equations)
GROUP 13. Boundary conditions and special sources
GROUP 14. Downstream pressure for PARAB=.TRUE.
GROUP 15. Termination of sweeps
GROUP 16. Termination of iterations
GROUP 17. Under-relaxation devices
GROUP 18. Limits on variables or increments to them
GROUP 19. Data communicated by satellite to GROUND
GROUP 20. Preliminary print-out
GROUP 21. Print-out of variables
GROUP 22. Spot-value print-out
GROUP 23. Field print-out and plot control
GROUP 24. Dumps for restarts
GROUP 25. VR-related settings
nogrid=t ! must appear below grdpwr
> DOM,    SIZE,        xulast, yvlast, zwlast !necessary to activate vre/v

store(prps)
real(posx,posy,posz,sizx,sizy,sizz) ! required declarations ?

> OBJ,    NAME,        ogeo:izz::ixx:
> OBJ,    POSITION,   :posx:,:posy:,:posz:
> OBJ,    SIZE,        xsizobj, ysizobj, zsizobj
> OBJ,    GEOMETRY,   :ogeo:
>obj,      type, blockage
>obj,     material,100
>obj,     grid, n
> OBJ,    ROTATION24,  orot
   GROUP 26  Graphical-display macros
STOP
</strong></pre></body></html>
```

The above blue-font inclusion is a PQ1 **template**, into which the PQ1 creator should supply whatever is appropriate to the now-to-be-created SimScene. He can also delete or modify any part of it.

It is divided into three parts. The first contains all the declared parameters and their default settings. The second calls, from the file 'frommenu.htm', whatever different settings the user may have entered subsequently in an interactive P-D menu session. The third makes all the consequential settings that constitute the essence of the SimScene which is in question.

Pq1ed provides other helpful macros. Suppose, for example, that the PQ1 creator wishes to introduce the character variable 'casename' into the 'general' group of parameters. Then, if below xml-group general he types ch thus :

**ch<Ctrl+j>**

there will appear on the screen:

```
char(?) ! description? = value  or, if list,
? = value1 !value2 !value3 !etc
```

whereupon he may decide to edit this as a list of options, thus:

```
char(casename) ! name of case
casename= case1 ! case2 ! case3
```

What choosing, one or other of these entails, of course depends on what he proposes to introduce into part 3 of the PQ1.

Similarly, typing **re**<**Ctrl+j**>, **in**<**Ctrl+j**>, or **bo**<**Ctrl+j**>, will provide the starting points for declaring and setting **re**al, **in**teger and **Bo**olean (*i.e.* logical) parameters.

**(c) Where macros are stored**

In the folder \phoenics\d_sapps\common\pq1ed, there resides a file called pq1ed.dci, which contains entries such as:

```
[ch | character declaration and setting]
char(?) ! description? = value  or, if list,
? = value1 !value2 !value3 !etc
```

and

[pq | PQ1 format]
\phoenics\d_sapps\common\pq1ed\macros\pq.txt

The entries are separated (and must be) by single blank lines.

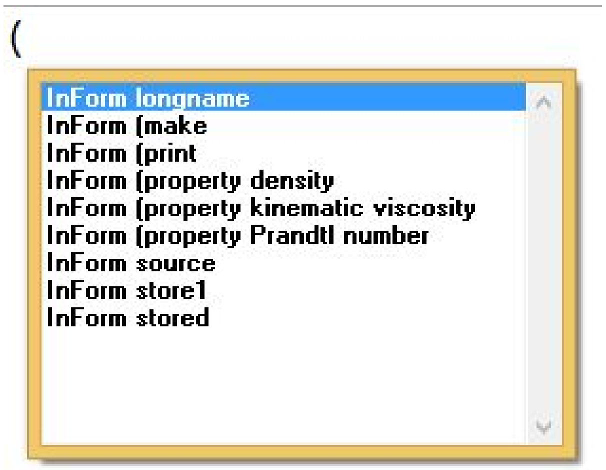
The first of the entries produced the above-described response to typing **ch**<**Ctrl+j**>. The second item produced the response to typing **pq**<**Ctrl+j**>.

The two entries differ in that the first is contained within pq1ed.dci itself, whereas, *viz.* the PQ1 template, is to be found in **pq.txt**, which is a file in the **\macros**\sub-directory.

Careful inspection, and copying, of these examples, enables pq1ed.dci and in-**\macros** files to be customized. Each PQ1-creator can therefore create precisely the macros which he needs. Indeed the 'other uses' mentioned above include the editing of Fortran or html  or other special-language files which require standardized inputs; for creation and attachment of appropriate macro files reduces typing time and the frequency of mistakes.

**(d) Code completion**

Macros allow both large and small already-prepared text sequences to be easily brought in and subsequently edited. Another equally valuable service is provided by what is called the 'code-completion' feature of Pq1ed. It is not restricted to, but is currently focused upon, providing helpful reminders of how PIL statements are properly written. Suppose, for example, that its user is aware that many In-Form statements begin with the bracket **(** followed by 'stored', or 'initial', or 'source'*, etc.,* but has forgotten the exact syntax. Then he can type <**Ctrl+space**> (
whereafter he sees immediately, in the file which he is editing, the following list:



```
(
InForm longname
InForm (make
InForm (print
InForm (property density
InForm (property kinematic viscosity
InForm (property Prandtl number
InForm source
InForm store1
InForm stored
```

Should it be '(print ' that he has in mind, then he may mouse-click on that line so as to select it, and then click on <**enter**>. Immediately will appear, in the file that he is editing

(print of STRING is EXPRESSION)

Other choices lead to;
(property den1 is FORMULA)

and to
(source of VAR3 at PNAME is FORMULA) .

These are reminders of how valid PIL lines are to be written. Here it is to be done by leaving the lower-case characters intact and replacing the upper-case ones by appropriate entries.

Just as macros respond to entries in the file called **pq1ed.dci**, so code-completion items respond to those in a file, *viz.* **pq1ed.pil**. This too is an ASCII file which is editable by the user. In it will be found the complete entries relating to PIL lines starting with an open round bracket, namely:

```
(make VAR1 is FORMULA) | In-Form's (make
(print of STRING is EXPRESSION) | InForm's (print
(property den1 is FORMULA) | In-Form's (property density
(property enul is FORMULA) | In-Form's (property kinematic viscosity
(property prndtl(VAR) is FORMULA) | In-Form's (property Prandtl number
(source of VAR3 at PNAME is FORMULA) | In-Form's (source
(store1 VAR3 is FORMULA) | In-Form's (store1
(stored VAR3 is FORMULA) | In-Form's (stored
```

Comparison of these lines with those just reported as appearing in the file under edit shows that that the characters to the right of **|** appear in the boxed to-be-selected list, whereas the characters to the left of it constitute the needed PIL entries themselves.

Not all the possible PIL statements appear in pq1ed.pil; and the explanatory words may be more meaningful to some readers than to others. This is why the customizability of pq1ed.pil is so welcome. Users can formulate the explanations in their own terms.

Although Pq1ed does what it can to make the PQ1-creator's task easy, it cannot dictate its further contents. The content of Part 3 of the PQ1 is for its creator to decide. When his work is finished, however, pq1ed can be asked to perform a very important task namely the creation of a complete SimScene allowing the PQ1's simulation capability to be activated. What this means is described at the end of section 2.2.

## 2.2 From pre-existing PQ1s

### (a) The task to be performed

There exist some PQ1s which are entirely satisfactory from the point of view of providing instructions to SATELLITE, but which cannot be presented to the user *via* SimScene menus. An example is core-library case 162, which concerns the release of gas into the atmosphere from a ruptured pipe-line. It is discussed in detail in the PHOENICS Encyclopaedia at: **/phoenics/d_polis/d_enc/enc_q1.htm#6**; and it ante-dates PHOENICS-Direct.

It does have an interactive-menu system; but only of the hard-to-use 'readvdu' kind, involving questions posed in a pre-set and unchangeable order. The task now to be discussed is how to enable the PQ1 to generate a SimScene-type menu.

The 'how?' question has a simple answer. Re-write declaration and value-setting of the existing PQ1 in the format displayed at the start of section 2.1 above. This is a mechanical task; so Pq1ed will in due course be taught to perform it automatically. At the time of writing (December, 2015), this has in part been accomplished, as is described below in section 3 (Further Capabilities) below. The task to be performed, whether so aided or by human labour alone, will now be described.

### (b) The declaration-and-settings part of case 162

This part of the library case now follows:

```
  !-------------------------------- declaration
```

```
   ! primary parameters, real
real(wallhigh,wallthck,walldist,indist,outdisnt)
real(domhigh,domwide)
real(pipezpos,relhigh,floorco)
real(pipediam,reldiam,relangl,gaspr)
real(windvel,windangl)        ! wind parameters
      ! secondary parameters
real(domlong,molwt,scale)
real(uwind,vwind)             ! horizontal wind components
real(xlen,xtot)
real(xp,yp,zp,xs,ys,zs,al,be,ga)
real(xpw,ypw,zpw,xsw,ysw,zsw) ! wall parameters for use in BOUND
real(rad)
real(valu)
   ! primary parameters, integer
integer(nxgrid,nygrid,nzgrid)
   ! secondary parameters
integer(nrx,ixwal1,ixwal2)
integer(intger)
integer(secno)
integer(nrz)
integer(io)   ! index of in-form objects
integer(iii)

   ! primary parameters, character
char(gas)                     ! what gas is released
char(tmodel,profl)            ! turbulence model; wind profile
   ! secondary parameters
char(uprofl,vprofl,gasden,gasvel,gasflo)
char(item,ch)

   ! primary parameters, boolean
boolean(pipe,wal,releas,diagnos,uniform,restart)
   ! secondary parameters
boolean(bool,subwal)

  !---------------------------------------- settings
        ! reals
        ! geometrical settings
wallhigh=2.5      ! used as a general length-scale parameter
scale=wallhigh    ! by means of this setting of scale
wallthck=0.2*scale ! which therefore influence all the rest

domhigh=3.5*scale
domwide= 12*scale

pipezpos=0.4*scale
indist= 2.5*scale
walldist=7.5*scale
outdist =14*scale

reldiam=2.0*0.0254  ! release diameter specified in inches
relhigh= 0.4*scale
relangl= 90 *3.14159/180     ! release angle; 0 = +ve x direction
floorco=0.005
pipediam=0.16*scale

windvel = 2
windangl=45   ! wind from north-west

   ! integers, defining the grid

nxgrid=50
nygrid=30
```

```
nzgrid=35

   ! characters
gas=h2s     ! hydrogen sulphide
gas=propane ! other possibilities are methane, ethane and propane
gaspr = 1.e5  ! i.e. one atmosphere

profl=(zg/:zwlast:)^(1./7.) ! one-seventh power law
profl=1.0                 ! uniform velocity
tmodel=lvel
tmodel=ke
   ! booleans
   ! theselogicals enable features of the simulation to be
   ! switched on or off, which permits their individual
   ! influences to be studied
releas=tA
wal=t

pipe=t
diagnos=f

uniform=f
restart=f
```

The rules for P-D-compatible PQ1s require that all parameters should be divided into groups; and each declaration should be individually made, and should be accompanied first by some words of explanation, and then its default setting. The template imported at the start of section 2.1 obeys those rules. The above extract from 162.htm, by contrast, declares many parameters **together**. These are acceptable to Satellite, but inconvenient for PHOENICS-Direct.

PQ1ed further requires, if it is to be told to create a SimScene, that Group names should appear thus:

**XML-Group Geometry**

Declaration and setting of parameters should appear one-by-one, each line containing a brief description as a comment, *i.e*. after an exclamation mark. The first of case 162 items thus should become:

```
  XML-Group Geometry
real(wallhigh) ! height of wall, m, used as a general length-scale parameter
wallhigh = 2.5
```

wherein the second line was extracted from
real(wallhigh,wallthck,walldist,indist,outdisnt)

and the third from
wallhigh=2.5 ! used as a general length-scale parameter.

The words 'height of wall,' above, were inserted by the human editor of the file; this implies that, even after Pq1ed has been taught to edit pre-existing PQ1s automatically, some additional human editing may prove desirable.

The same laborious but conceptually trivial one-parameter-at-a-time re-writing must now be undertaken for all the parameters of the original PQ1. When performed automatically, the xml-grouping will accord with the 'settings' section rather than with the real-integer-character-boolean ordering of the 'declarations' section. A later-coming human editor may prefer a different grouping; which he can arrange *via* simple cut-and-paste operations.

The final line of this part of the edited PQ1 will be:

```
  XML-End
```

with two spaces preceding the X.

**(b) The menu**

File 162.htm also contains its own interactive section, which PD does not need. Instead, as part 2 of the interactive section (*Question and answer*), only one line should be added, namely:

incl(frommenu.htm)

This brings in whatever settings the user will have made by way of the PHOENICS-Direct-provided menu. This, while assisting users easily to see and modify the relevant parameters also connects with:

- a **descr_en.htm** file, in the SimScene's DOCS folder, which describes the scenario to be simulated and provides other advice which is helpful to the user (Note that '_en' indicates 'English'(*i.e.* the language in which the scenario is written), as would '_ru' 'Russian', '_ja' Japanese, *etc.,* the language-specific character sets of which PHOENICS-Direct can handle, when so instructed);

- the PQ1 file itself, as **q1.htm**, in the SimScene's INPUT folder; and

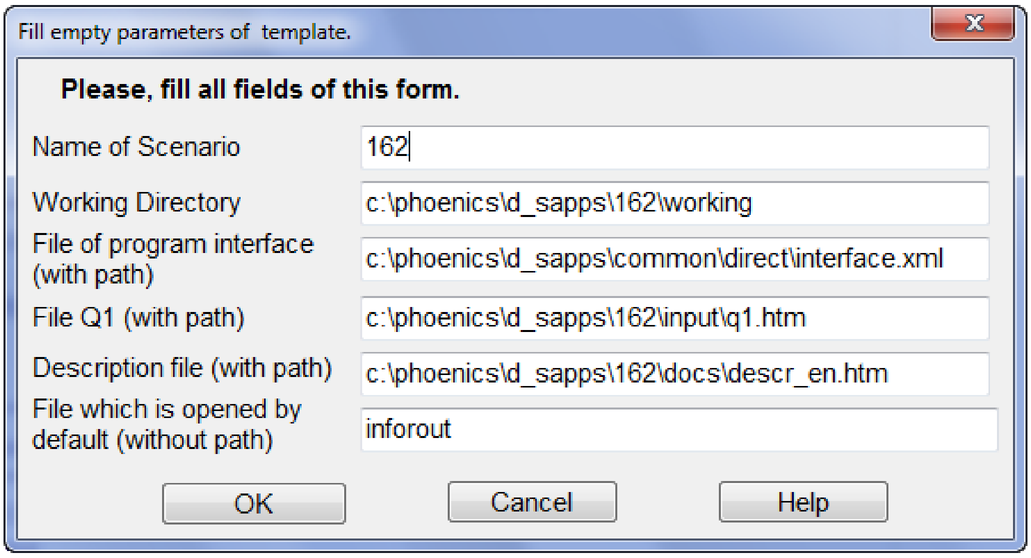
- a **scene.XML** file, also in the INPUT folder, which re-formats part of the content of the PQ1 and allows its parameter settings to be changed by way of a menu.

The PQ1 is always an ASCII file, because it must be interpreted by the English-only PHOENICS Satellite module; but the other two should have utf-8 format, if Russian, Japanese, Chinese or other character sets are being employed.

**(d) Creating the SimScene**

Of course, the whole system of folders and files constituting a SimScene can be created manually; and, if they exist, they can be simply updated. Alternatively, if they do not, or if it is preferred to create a new one, one of Pq1ed's most valuable functions can be activated. This is its ability to create a complete SimScene, instructed by a single click of the user's mouse after he has pressed the **F2** key on the keyboard so as to save the file. This causes a menu to be presented, of which he should choose: '**Create SimScene**'**.**

The following new window '**Fill empty parameters of template**' then appears. In this the user has to print only the name of the scenario; for the other fields will be filled automatically. The window will then look as follows.



Fill empty parameters of template.

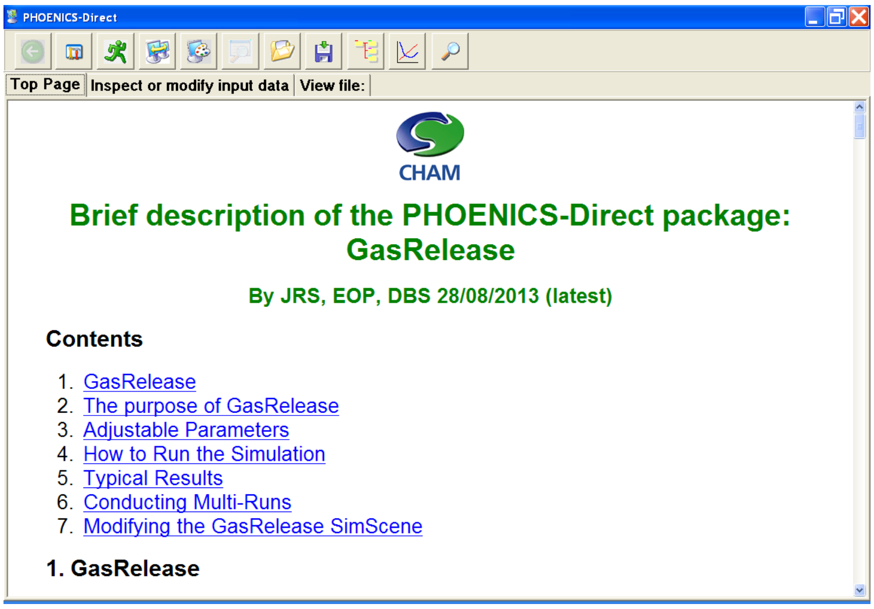
**Please, fill all fields of this form.**

| | |
|---|---|
| Name of Scenario | 162 |
| Working Directory | c:\phoenics\d_sapps\162\working |
| File of program interface (with path) | c:\phoenics\d_sapps\common\direct\interface.xml |
| File Q1 (with path) | c:\phoenics\d_sapps\162\input\q1.htm |
| Description file (with path) | c:\phoenics\d_sapps\162\docs\descr_en.htm |
| File which is opened by default (without path) | inforout |

Pressing the **OK** button will then create:

- directories of a new scenario,
- a template of the description file **c:\phoenics\d_sapps\162\docs\descr_en.htm**;
- the scenario file **c:\phoenics\d_sapps\162\input\ scene.xml**;
- the file **c:\phoenics\d_sapps\162\input\q1.htm**, which is a copy of the just-edited PQ1; and
- the batch file **c:\phoenics\d_sapps\162\start.bat**, which launches the SimScene.

All these actions have in fact been taken, with the differences that the name GasRelease has been chosen for the SimScene in place of the less informative 162, and that the template for descr_en.htm has been replaced by an informative GasRelease-specific document. Its top page is shown below.



**Brief description of the PHOENICS-Direct package: GasRelease**

**By JRS, EOP, DBS 28/08/2013 (latest)**

**Contents**

1. GasRelease
2. The purpose of GasRelease
3. Adjustable Parameters
4. How to Run the Simulation
5. Typical Results
6. Conducting Multi-Runs
7. Modifying the GasRelease SimScene

**1. GasRelease**

This needs not be the end of Pq1ed's interest, it should be remarked, for SimScenes are always open to improvements, which must be introduced by editing the PQ1 which now is embodied in: \input\q1.htm). At the end of editing, pressing F2 will enable File >Update 'SimScene' to be selected, which automatically updates scene.xml, so as to correspond to the new q1.htm.

The logically-entailed changes to **descr_en.htm**, on the other hand, must of course be introduced by a human editor.

## 2.3 From pre-existing single-instance Q1s

There are two main reasons for wishing to convert single-instance Q1s into parameterized form. First, there exist hundreds of **moderately** interesting cases in the PHOENICS Input-File libraries which would become **very** interesting if only their users were enabled easily to vary the physical or numerical input data either by simple hand-editing or, better still, by way of a SimScene-type menu.

Secondly, single-instance Q1s created by the VR-Editor for large-scale CFD simulations of practical importance are very often of the 'worst-of-both-worlds' character, namely having grids which, for the solid objects in the scenario, are too fine for economy and too coarse for accuracy. PHOENICS does possess means of alleviating these disadvantages; but these can be introduced into non-parameterized Q1s only with such difficulty that they are almost never employed.

Thought was first given to whether Pq1ed should be empowered to perform the task of creating a PQ1 on its own, starting only from the given Q1 and its Q1EAR, EARDAT and FACETDAT derivatives. This however was judged to be unnecessarily difficult because the PHOENICS Satellite module can be taught to express its knowledge of its user's settings directly in parameterized format.

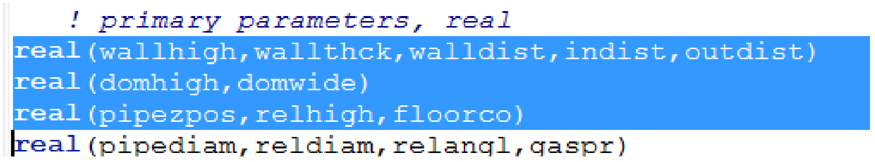
The recent introduction into the PHOENCS Input language of the new OBJINFO command has now made this possible. Its exploitation by the not-yet-existent PQ1Maker module remains a task for the future.

## 3. Further capabilities

### 3.1. Automatic creation of scenario

Pq1ed has been provided with another way of editing parameterized Q1 files into the format fitted for automatic creation of **scene.xml**.

The user should select one or several lines where variables are declared, *e.g.* as the next lines show:



```
   ! primary parameters, real
real(wallhigh,wallthck,walldist,indist,outdist)
real(domhigh,domwide)
real(pipezpos,relhigh,floorco)
real(pipediam,reldiam,relangl,gaspr)
```

It is important, when selecting the lines, to place the cursor at the **beginning** of next line, in the given example – at the beginning of the fourth line. If the cursor is placed at the **end** of the third line, only two first lines will be converted.

Then, upon choosing **Edit>**'**Conversion to XML**', the selected lines will be changed in to the following:

```
real(wallhigh,wallthck,walldist,indist,outdist)
real(domhigh,domwide)
real(pipezpos,relhigh,floorco)
  XML-Group
real(wallhigh)   !
wallhigh =
real(wallthck)   !
wallthck =
real(walldist)   !
walldist =
real(indist)   !
indist =
real(outdist)   !
outdist =
real(domhigh)   !
domhigh =
real(domwide)   !
domwide =
real(pipezpos)   !
pipezpos =

real(relhigh)   !
relhigh =
real(floorco)   !
floorco =
  XML-End
incl()
```

Earlier selected lines do remain, to show to the user that this conversion has been made correctly; but all declared variables have been written in the PQ1Editor format for further editing of **scene.xml**. The user should verify the formatting accuracy, introduce the name of the group and all variables, then add parameter values and the include-filename, if such a file is required. Only after that should the extra lines be removed.

Upon editing, the indicated fragment looks as follows:

```
  XML-Group Geometry [geom.htm]
real(wallhigh)   ! height of the wall, m
wallhigh = 2.5
real(wallthck)   ! thickness of the wall, m
wallthck = 0.2*wallhigh
real(walldist)   ! distance from hole to the wall, m
walldist=7.5*wallhigh
real(indist)   ! distance from inlet to the hole, m
indist = 2.5*wallhigh
real(outdist)   ! distance from wall to the outlet, m
outdist = 14*wallhigh
real(domhigh)   ! height of the domain, m
domhigh = 3.5*wallhigh
real(domwide)   ! width of the domain, m
domwide = 12*wallhigh
real(pipezpos)   ! position of pipe, m
pipezpos=0.4*wallhigh
real(relhigh)   ! release high, m
relhigh= 0.4*wallhigh
real(floorco)   ! friction factor on floor
floorco = 0.01
  XML-End
incl(geom.htm)
```

It should be noted that:

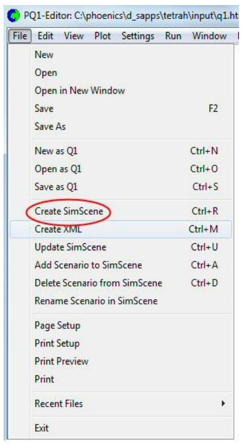
- parameters of the given group will be saved in the file **geom.htm**, not in **frommenu.htm**.
- Description of parameter groups can be located in any place of Q1 file, the only thing required is to type the line **XML-End** at the end of the text referring to the xml-groups.
- Declaration of group parameters can be made not only in PQ1 but in any file which is connected to Q1 by the operator **incl( ).**

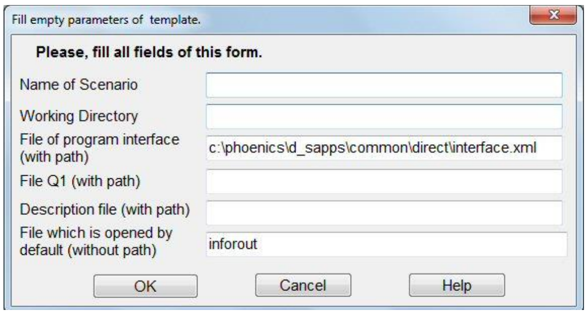
Upon creation of directories and scenario files, several first parameters of this group will be written in the file **scene.xml** as is shown below:

```
<group visible="yes">
  <name>geometry</name>
  <file>geom.htm</file>
  <param visible="yes">
    <name>height of the wall, m</name>
    <variable>wallhigh</variable>
    <variable_type>real</variable_type>
    <declared>yes</declared>
    <value>2.5</value>
  </param>
  <param visible="yes">
    <name>thickness of the wall, m</name>
    <variable>wallthck</variable>
    <variable_type>real</variable_type>
    <declared>yes</declared>
    <value>0.2*wallhigh</value>
  </param>
  <param visible="yes">
    <name>distance from hole to the wall, m</name>
    <variable>walldist</variable>
    <variable_type>real</variable_type>
    <declared>yes</declared>
    <value>7.5*wallhigh</value>
  </param>
```

## 3.2. The Multi-scenario SimScene possibility

A new opportunity that has been recently introduced into Pq1ed is a possibility of creating SimScenes wherein several separate scenarios are combined. If, for instance, there are 3 PQ1s that we want to combine into one SimScene, then the algorithm to do this is as follows.
1. Open the first Q1 in its folder in the Pq1ed and create on its basis a new scenario in the usual way. You will then be able to create a scenario *via* command **Create SimScene** which can be inactive.But it will become active when you do the following:
Click the command **Create SimScene**, whereafter the message “Variable declarations are checked successfully” will appear. Click “OK” and the fill-in scenario box will appear.

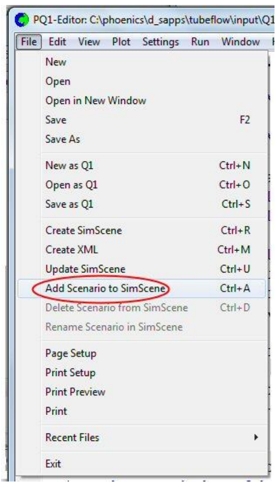



Fill empty parameters of template.

**Please, fill all fields of this form.**

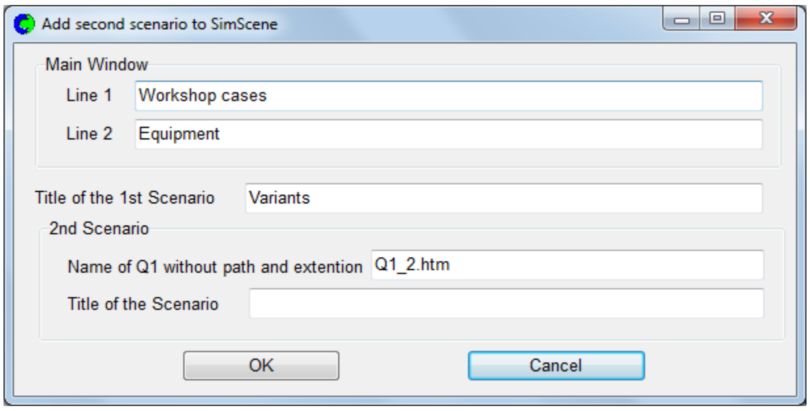
| | |
|---|---|
| Name of Scenario | |
| Working Directory | |
| File of program interface (with path) | c:\phoenics\d_sapps\common\direct\interface.xml |
| File Q1 (with path) | |
| Description file (with path) | |
| File which is opened by default (without path) | inforout |

2. Introduce the new multi-scenario SimScene name, *e.g.* **Variants**. Then click OK.
3. There will be the message “Scenario file is created successfully”, click OK.
4. You can at this stage check **d_sapps** directory and find there the folder **Variants** containing a usual collection of folders for any SimScene. But this SimScene does contain only one scenario created on the basis of the 1st Q1 file. Let’s add another one to it.
5. Open the second Q1 in its folder via PQ1 Editor paying attention that it is understood by the editor as a Q1 file, otherwise it might be necessary to resave it as a Q1 file.
6. Go to **File – Add Scenario to SimScene**. It will then be necessary to choose the earlier-



created **scene.xml** file from **d_sapps\variants\input** folder.

7. Message “Variable declarations are checked successfully” will be displayed. Click OK.
8. The window “Add second scenario to SimScene” that follows will appear.


Add second scenario to SimScene

Main Window

| | |
|---|---|
| Line 1 | Workshop cases |
| Line 2 | Equipment |
| Title of the 1st Scenario | Variants |

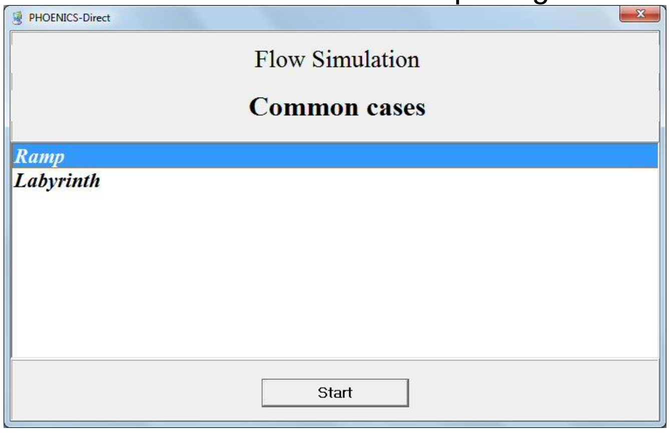
2nd Scenario

| | |
|---|---|
| Name of Q1 without path and extention | Q1_2.htm |
| Title of the Scenario | |

In these window boxes the user should type the information which will later appear in the SimScene Variants opening window.
Let us now type in the Line 1 box instead of “Workshop cases” a more meaningful title, *e.g.* “Flow Simulation” and in the second line box “Common cases” instead of “Equipment”.
The first scenario title will be, *e.g.*, “Ramp”, and the second one –

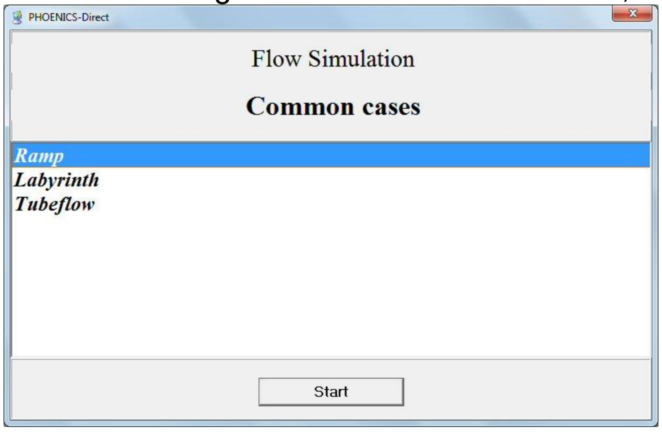
"Labyrinth". The user can choose any titles and explaining information that he would like to display in the opening window and click OK.
Note that the name of the second scenario by default will be **Q1_2.htm** (see the box "Name of Q1 without path and extension") to distinguish it from the first scenario copied to the **Input** folder under its original name, **q1.htm**. However, the user can save the second scenario under any name.

9. There will be the message that "Scenario file is added successfully". Click OK.
10. Let us now see what we obtained in the end. Open the **Variants** folder and run this scenario via **start.bat**. The opening window should look as follows.



You may now choose any case, Ramp or Labyrinth, to work with later and press the Start button. Each selected scenario will be opened with its own menu and the user can work with any of them.

11. Now, if another scenario should be added to this Variants SimScene, the items 5-7 of this algorithm should be repeated and its new title should be given. The third Q1 will be copied to the **Variants\input** folder as **Q1_3.htm** and so on.
12. After adding the third Tubeflow scenario, the opening window will look as follows.



13. It is also possible to delete or rename any scenario in the multi-scenario SimScene via commands **File – Delete Scenario from SimScene** or **File – Rename Scenario**

## 3.3 Treatment of INCLuded files

The INCLude command of PIL is a long-standing feature providing the advantage that files of general usefulness, such as those in **\d_object\pilobj** can be imported into many different PQ1s at the moment when Satellite reads them.

When the Pq1editor encounters INCL statements, it too acts as though the contents of the INCuded files were actually present in the PQ1; and it does so in whichever of its two modes of operation it is being used, these being:

(a) **editorial**, *i.e*. enabling textual changes to be made; and

(b) **creative**, *i.e*. writing corresponding-to-pq1 scene.xmls and other SimScene-related files.

In respect of the editorial mode (a), let us take the ***incl(frommenu.htm)*** statement as an example, and specifically that of the Ramp SimScene. On reading its INCL statement, PQ1ed will present the following on the screen:

```
!------- original line (incl(frommenu.htm)) PLEASE, DO NOT EDIT OR REMOVE THIS LINE !!!
!------- start of included file (C:\phoenics\d_sapps\ramp1\input\frommenu.htm) PLEASE, DO NOT EDIT OR REMOVE THIS LINE !!!
```

followed by:

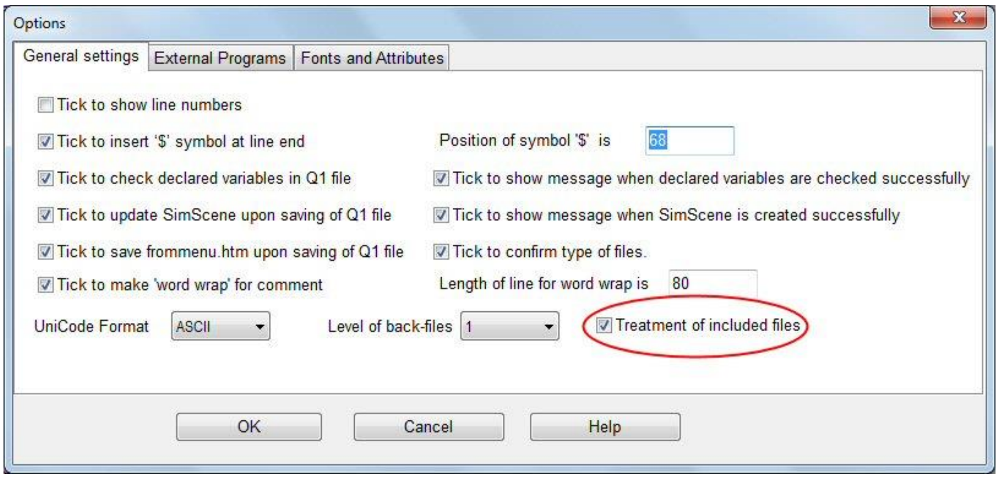
```
 <html><head><title>frommenu.htm</title>
 <link rel="stylesheet" type="text/css"
  href="/phoenics/d_polis/polstyle.css">
 </head><body><pre><strong>
 ! Group grid
nxx = 20 ! number of cells in x-direction
nyy = 20 ! number of cells in y-direction
nzz = 20 ! number of cells in z-direction
sweepn = 100 ! number of iterations

 ! Group geometry
flolen = 60. ! length of floor, m
flowid = 15. ! width of floor, m
ceilhigh = 3. ! height of ceiling, m
flooth = 0.5 ! thickness of floor, m
applen = 20. ! apperture length, m
appwid = 5. ! apperture width, m

 ! Group turbulence and flow
turb = t ! turbulence
flowdir = positivez ! select flow direction (positive or negative)
 </strong></pre></body></html>
```

and then:

```
!------- end of included file (C:\phoenics\d_sapps\ramp1\input\frommenu.htm) PLEASE, DO NOT EDIT OR REMOVE THIS LINE !!!
```

The lines printed here in blue font represent the actual content of the frommenu.htm. The lines printed in green font are messages sent both to itself and to the user. To itself they say: "Here are the beginnings and ends of the included file". They are needed as markers; hence the "PLEASE DO NOT EDIT" is addressed to the user; to which might have been helpfully added: "but you MAY edit or remove anything **within** the included material, just a though you had loaded the file into the editor yourself".

The display of the contents of the included file is a consequence of the default setting in Settings – Options – Treatment of included files*, i.e.*


Options

General settings | External Programs | Fonts and Attributes

[ ] Tick to show line numbers

[✓] Tick to insert '$' symbol at line end — Position of symbol '$' is 68

[✓] Tick to check declared variables in Q1 file — [✓] Tick to show message when declared variables are checked successfully

[✓] Tick to update SimScene upon saving of Q1 file — [✓] Tick to show message when SimScene is created successfully

[✓] Tick to save frommenu.htm upon saving of Q1 file — [✓] Tick to confirm type of files.

[✓] Tick to make 'word wrap' for comment — Length of line for word wrap is 80

UniCode Format: ASCII — Level of back-files: 1 — [✓] Treatment of included files

Clearing the box in question will result in the following: only the INCL statement **incl(frommenu.htm)** will be left and the contents of the included file will be hidden. Only if this box is ticked, it is possible to edit the contents of the included file.

No such ticking is needed to cause PQ1ed to perform its second function (b). When the user wishes to create the scene.xml which corresponds to the Q1 contents, he does so *via* commands **File – Create Simscene** or **File – Update Simscene,** whether the Q1 does or does not contains INCL statements.

## 4. Concluding Remarks

The PQ1 editor is a utility which is being continually developed, to keep pace with the developments in PHOENICS and in the manner in which it is used. For example, current attention is being devoted to the various manifestations of the 'Connected Multi-Run' concept; which may require extension of the scene.xml format. When these developments have attained a more settled state, further versions of PQ1ed are to be expected.