SALOME 3.2.0

Major release announcement

May 2006

General information

OPEN CASCADE is pleased to announce major SALOME 3.2.0 version. It is a maintenance release that contains the results of planned major and minor improvements and bug fixes with reference to the SALOME 2.2.2 version released at the beginning of year 2005.

TABLE OF CONTENTS

New features and improvements

Installation wizard

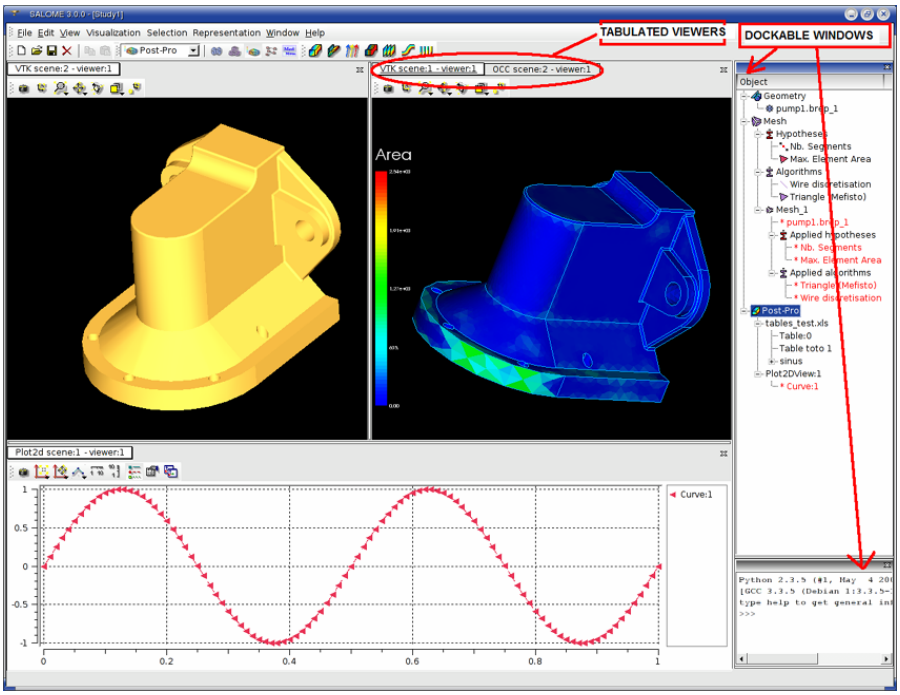
- From version 3.2.0 the installation procedure supports four different Linux distributions and provides binaries compiled specially for the following OS:
  - Debian Sarge
  - Red Hat 8 (Red Hat 9 compatible binaries also)
  - Mandrake 10.1 (RedHat Enterprise 4 compatible binaries)
  - Mandriva 2006

  Please refer to chapter “Supported Linux distributions” for more details.
- From version 3.2.0 the installation procedure generates an additional folder in the user's home directory. This folder includes scripts which allow launching SALOME servers in different modes according to the Application notion introduced by EDF (please refer to KERNEL services documentation). The name of this folder is generated according to the number of the installed SALOME version: for example, for version 3.2.0 the folder name is salome_appli_3.2.0. The environment script which is required for SALOME launching is also automatically generated in this folder by the installation procedure.
  NOTE: this folder is created only if the user installs KERNEL module binaries package.
- Debian installation procedure forces installation of all products from binaries as a default option.
- Compilation of SALOME from sources function has been added. Now it’s possible to choose compilation of SALOME from sources.
- Compilation of Open Cascade Technology from sources has been added.
- Unnecessary products (happy doc) have been removed from install proc.

General modifications

New look and feel

SALOME look-and-feel has been significantly improved. Please find below precise explanations concerning all changes and improvements.



Specific GUI style

Special own GUI style has been developed for SALOME. The style differs from standard KDE styles in the following aspects:

- New menu design; Gradient fill in menus and toolbars;
- Non-active buttons displayed in transparent grey scale;
- New and simple look for all controls;
- Buttons are highlighted when the mouse cursor points at them;
- Current dockable window has active header;
- Gradient fill in the headers of QlistView class;
- New look of splitter in class QSplitter;
- New look of tabs in class QTabWidget.

Multi-desktop approach

A Multi-desktop GUI is now in SALOME. In fact, each document (Study) has a separate Desktop. Users feel more comfortable and will not be confused because now all windows connected to a certain Study (viewers, dockable windows etc) are placed in one container. This approach is much better than the classic multi-document interface in case if many windows are open, i.e. if one document contains a lot of viewers, dialogs, etc. Each desktop also has its own menu and toolbar.

Dockable toolbars and panels

In SALOME 2.x.x the desktop space was distributed among windows through a special container – splitter. It allowed changing the sizes of the windows on the desktop. Since SALOME 3.2.0 almost all windows have been redesigned to use the dockable mechanism. Dockable windows and toolbars can be placed in special areas, on the borders of the desktop. The size and position of dockable windows can be changed by the user. He can even place such windows outside the desktop or hide them. Dockable windows are Object Browser, Python Console, MenuBar, etc.

Tabbed viewers

There is a new window manager in SALOME 3.2.0. As opposed to the manager from 2.x.x version, the new one displays only one active window in the representation area. All windows are placed inside tabs where the user can switch from one to another. If there are several windows and the user wants to display more then one at the same time, he can split the representation area in two parts, in the horizontal or vertical direction; this action creates two representation areas, instead of one, with windows placed in them. The size of such representation areas can be changed by the user. He can also move any windows between representation areas. When an area becomes empty, it is not displayed.

New user preferences GUI (menu File / Preferences)

A new mechanism for working with preferences and resources is introduced in SALOME 3.2.0. Now the editor of preferences is common for all modules and it uses only one dialog box where users can customize preferences for each module. When the user changes the setting (clicks the “OK” button) each module is notified about the preferences that have been changed. The users can also edit preferences in module-specific windows.

Automated saving of look and feel preferences

In 3.2.0 version all positions and sizes of dockable windows are stored in a special user file. Each time the user quits the application, the window coordinates are automatically saved in this file to be restored at the next start of the application. There are also differences in the settings of the same windows (for example, Python console) between different modules. It means that the same window can have different size and position when the other module is activated. Each module can also indicate which common windows are necessary for working, so that only they could be displayed when this module is active.

Light components (engine-independent)

Thanks to the architectural changes in the SALOME GUI, SALOME 3.2.0 version enables to create engine-independent components. These components may not use CORBA at all, having the internal data structure, which can be written in pure C++ (or python). Such modules are located inside the SALOME GUI process and from the end user point of view does not differ from standard components. The “LIGHT” component, an example of such components, is placed into EXAMPLES CVS. Please refer to chapter about KERNEL light configuration.

Dump study (dump Python)

A special feature to create a python script which can rebuild all data from a SALOME study has been implemented. Now a study created by the GUI can be written into a python script for automation. This feature is supported in GEOM, SMESH, VISU and SUPERV modules and can be extended to any module based on the SALOME engine.

Dependency on MED library removed from KERNEL

MED Wrapper library has been moved from KERNEL to MED module. Thus, SALOME KERNEL no longer depends on the MED library. This change has also affected other SALOME modules. In particular, configuration procedures have been updated for standard SALOME modules, which use the MED library. They detect the MED library presence themselves, rather than relying on KERNEL means as it was done previously.

Examples data files movement

Examples data files from MODULE_ROOT_DIR/examples have been regrouped in a separate folder according to file types. Corresponding CVS module SAMPLES_SRC is created under the CVSROOT EXAMPLES. Its structure is the following:

SAMPLES_SRC – root directory
SAMPLES_SRC/MedFiles – subdirectory for MED files
SAMPLES_SRC/Unv – subdirectory for unv files
SAMPLES_SRC/Sauv – subdirectory for sauv files
SAMPLES_SRC/inp_xyz – subdirectory for inp and xyz from MED module
SAMPLES_SRC/Shapes – subdirectory for geom shapes (in BREP, IGES and STEP formats)
SAMPLES_SRC/Shapes/Brep – subdirectory for BREP files
SAMPLES_SRC/Shapes/Iges – subdirectory for IGES files
SAMPLES_SRC/Shapes/Step – subdirectory for STEP files
SAMPLES_SRC/Tables – subdirectory for VISU table files
SAMPLES_SRC/Superv/ – subdirectory for supervisor example files
SAMPLES_SRC/Superv/Python – subdirectory for python scripts
SAMPLES_SRC/Superv/Graphs – subdirectory for XML files which contains graphs for Supervision

The corresponding files have been removed from other SALOME modules except for the MED component. Files in the MED_ROOT_DIR/resources were left in order to support current test mechanism for the MED component.

A new environment variable DATA_DIR has been introduced to indicate the location of SAMPLES_SRC. All corresponded python file examples were updated to support new placement of data files. All TUI tests were updated too.

Porting

Salome has been ported to Mandriva2006 and Mandriva64 as well. As Mandriva64 is a 64 bit platform, SALOME VTK functionality has been ported to VTK 4.4. Consequently, in order to support both versions of VTK (current 4.2.6 and new 4.4) we need to use vtkFloatingPointType instead of bare float point type for all VTK functionalities.
KERNEL, MED and COMPONENT (part of EXAMPLES) have been changed to be portable to CCRT machine (DEC OSF ALFA).

Documentation

Context help has been added in most dialog boxes in all modules. Documentation was updated to present state of functionality. Also currently it's possible to generate doxygen sources documentation using "make dev_docs" step.

Version system improvements

SALOME versioning system improvement was implemented. It has been done for all the SALOME modules - KERNEL, GUI, MED, GEOM, SMESH, VISU, SUPERV, mesh plugins modules and all samples modules. The improvements include the following:

- Remove duplication of version number in different files. The version number is now defined only in configure.in/configure.ac file. All other files (like VERSION, etc) are generated from the corresponding *.in template by configure script (via macro-substitution).
- The file configure.in/configure.ac file defines two variables: VERSION (string representation of version number) and XVERSION (hexadecimal version number).
- For each SALOME module an additional <MODULE>_version.h file is implemented which can be used to check module version at the compilation step (except Python modules, like PYCALCULATOR, PYHELLO, etc). For example, for the KERNEL module the file name is KERNEL_version.h.
- KERNEL module now includes salome_version.py Python script which provides methods to get version number for any SALOME module.

KERNEL improvements

Separation and repackaging

The KERNEL module has been repackaged for 3.2.0 version and from now KERNEL and the GUI are separated. KERNEL provides basic functionalities for Container and Component management, SALOME Data Structure implementation, Life Cycle mechanism etc. The GUI module deals with definition of interactive services, i.e. Qt written application (desktop, basic dialogs etc), viewers (VTK, Open CASCADE Technology, etc.), selection mechanism etc. It allows writing pure batch components which use only the KERNEL module, for example, for batch mode parallel computations.

SALOME DS proxy

A concept of SALOMEDS proxy has been implemented in Salome version 3.2.0 in order to optimize the usage of SALOMEDS services and facilitate the development of GUI components without CORBA. The modifications group in two main parts:

- Repackaging of the SALOMEDS package. The SALOMEDS package has been divided in two 2 packages: SALOMEDSImpl - this package contains a pure C++ (no CORBA involved) implementation of SALOMEDS services and SALOMEDS - the package containing CORBA wrappers to the services implemented in the package SALOMEDSImpl
- Creation of special wrappers, which can be used for calling SALOMEDS services. The wrappers are declared as abstract C++ interface classes, whose structures and declarations coincide with the interfaces given in the IDL description of SALOMEDS.

These abstract interfaces are presently implemented for SALOMEDS objects. All calls to SALOMEDS are hidden in the interface.

Hiding the calls to the real implementation allows the developer to increase the performance while using the SALOMEDS services in case when the SALOMEDS server and its clients are running in the same address space. In this case all calls to the SALOMEDS services are directed to the local implantation in SALOMEDSImpl without involving the CORBA mechanism.

KERNEL migration on automake-autoconf tools

KERNEL development environment was changed in order to provide usage of automake-autoconf GNU tools.
This modification caused some important changes in configuration procedure of KERNEL:

- Configuration on RED HAT 8 has become available only basing on the latest autoconf-automake prerequisites.
  - The other way to configure is to build an archive of sources on Mandrake after build_configure with the help of “make dist” command and then use these archives on REDHAT 8 (don’t call build_configure, but start from configure step in this way)
    Note: In the Installation wizard there is such prepared archive and user can compile sources using standard “build.csh” script from installation procedure.
- Now install is an obligatory step for creation of binaries

Simplification of SALOME configuration scripts

Recent separation of the KERNEL module into two independent entities: KERNEL and GUI has caused a change of dependencies on the SALOME pre-requisites. Sometimes there is a need of CORBA-independent modules (SALOME light modules). Naturally these modules may need to be independent from KERNEL-specific pre-requisites such as omniORB or may require only partial use of KERNEL or other standard modules.
To reflect these changes a new configuration option has been introduced allowing compiling SALOME light modules using only GUI package and KERNEL light configuration.

KERNEL light configuration.

KERNEL light configuration includes pure C++ packages (without CORBA staff) which are necessary for other modules:

- HDFpersist
- CASCatch
- Basics
- SALOMELocalTrace

A new key “–disable-corba-gen” is added to the KERNEL module with the following behavior:

- This key is optional by default (–enable-light=no)
- Light configuration is active, if the following configuration options are used:
- --disable-corba-gen | --disable-croba-gen =yes

The files configure.in.base and build_configure have been modified.
Switching to this configuration, KERNEL will be built only including pure C++ modules (see above).

Configuration procedure

- build_configure does not anymore need to be invoked from the source directory.
- A lot of no longer relevant files have been removed from sources.

SALOME Application

- Improvement of cleaning process when the application is killed.

Launch procedure

- SalomeApp.xml is now split among all modules (see GUI modifications for more details);
- runSalome support "—modules" option has been restored.

Developer and integrator documentation (in source tarball)

- doc directory contains several documents in progress:
- SALOME Application : how to install and use the SALOME application on several computers;
- Unit Tests : organization of sources, unit tests procedures;
- Kernel Resources: some tools for developers (other documents are obsolete).
- User TUI documentation updated.
- The possibility to generate documentation from Python source by DOXYGEN has been added.

SALOME_Comm package (efficient distributed memory data transfer)

- A matrix interface has been added.

CORBA transfer service for large files

There is a C++ class (SALOME_FileTransferCORBA) which provides a transfer service for copies of large files, given a distant computer hostname and a file path on this computer. This service relies on container servers (LifeCycleCORBA). The service keeps trace of copies already edited on a computer, to avoid multiple copies (and if the reference file is on the local computer, no copies are created). It is also possible to share an object representing the file between CORBA servers and client, to obtain local copies.
The interface is available in Python via Swig.

Unification of Python scripts used either inside GUI server or outside it (batch mode)

A SALOME Session can be launched with or without GUI (with a Python interpreter interface only, see doc/SALOME_Application) and the Python interpreter can be launched outside an already running SALOME session and connected to it.
Provided a Python script does not use the Swig interface from GUI, it can be used inside or outside the GUI server. For instance, scripts based on geompy.py or smesh.py work with external interpreters.
Consequently the non regression test base has been updated to use geom.py and visu.py, SMESH tests are in progress.

Unit Tests

- Python Unit testing capability has been added, it tests application written on LifeCycleCORBA, and also tests Python Swig interface.

Other KERNEL improvements

- SALOME 3.2.0 includes major modifications including a redesign of message traces and introduction of unitary tests based on the cppunit utility.
- Local code branches in client classes have been protected with SALOMEDS::Locker to make local calls to SALOMEDS via SALOMEDSClient classes thread-safe when STUDY is embedded.
- Problem with the module name displayed in lower case has been fixed.
- If the user does not have permissions to save study, a warning is displayed properly.
- DumpStudy() method has been debugged.
- 'Invalid pointer' message when loading GEOM module after restoring the study from HDF-file has been fixed.

GUI module

Customization of viewers for modules

Existing viewers (Open Cascade Technology viewer 3d, VTK viewer 3d, Qwt plot, etc) have been separated into different libraries. All of them are written using the base interface and have a similar structure. Each module can ask the application to create a copy of a predefined viewer or can add its own viewer inherited from the base interface into the application.

New GL accelerated 2D viewer

As one of the standard components of SALOME GUI, now there is a new type of viewer. This is special 2D viewer used OpenGL for fast visualization of 2D primitives. In spite of it is not used in standard modules, it is used in other custom modules to visualize 2D schemas with high performance.

GUI of modules as plug-ins

All GUI of standard modules in SALOME 3.2.0 have a plug-in structure, i.e. each module is placed in the dynamic library which is loaded on demand. All modules can create their own menu items, buttons in toolbars, windows etc.

Ability to build non-SALOME applications (CAM level)

There is a special separate CAM level, used for writing modules which don't use standard tools of the SALOME platform. The CAM component serves as a basis for the new SALOME GUI and contains all basic functionality for working with modules (loading a module; opening, saving, closing a document, customization of toolbars and menus).

GUI light configuration

GUI light configuration will be used only with light configured KERNEL.
Some configuration check files have been moved from KERNEL to GUI package:

- check_vtk.m4
- check_qwt.m4
- check_sip.m4
- check_msg2qm.m4
- check_pyqt.m4

Some new check files have been added:

- check_disable_Corba.m4
- check_GLViewer.m4
- check_OCCViewer.m4
- check_Plot2dViewer.m4
- check_PyConsole.m4
- check_SalomeObject.m4
- check_SupervGraphViewer.m4
- check_VTKViewer.m4.

A new key --disable-corba-gen has been implemented. By default it is set to "no".
This flag can be used for setting light configuration (without Corba and with KERNEL light configured module as prerequisite).

Packages, which have dependency from CORBA, have not been compiled and used in this case.
It is possible to sets another flags with this one:

- --disable-pyConsole by default – “no”
- --disable-glViewer by default – “no”
- --disable-plot2dViewer by default – “no”
- --disable-supervGraphViewer by default – “no”
- --disable-occViewer by default – “no”
- --disable-vtkViewer by default – “no”
- --disable-salomeObject by default – “no”.

HDF persistence for light configuration

Now GUI light configuration of bases on HDFPersist classes from KERNEL light configuration. This modification allows having transparent persistence in one common HDF format for both “full” and “light” components.

Resource files modification

- From now on, we have SalomeApp.xml configuration file not only in GUI module - each SALOME component (except mesh plug-ins) has its own copy of this file. So, if you add or modify some preferences for a certain component, please, put them in the corresponding SalomeApp.xml file. It is not allowed to put such component-related stuff in the general SalomeApp.xml (which is in GUI module). Those who develop custom SALOME-based modules, have to create their own SalomeApp.xml files and put all preferences there.

- In addition, user preference files from the nearest version (.SalomeApprc_version_name) are imported automatically on the first run of this version of SALOME.

SIGFPE problem on some graphic card configuration (standard NVIDIA drivers on Mandrake 10.1)

Actually the problem existed before version 3.2.0 but it was not visible, since the FPE signals were not raised. The matter is that recently the OCT function OSD::SetSignal(Standard_Boolean aFloatingSignal) has been corrected so that it could raise FPE signal on Linux when an arithmetic exception occurs (the system function feenableexcept is used there).
To have this bug fixed we have updated the call to OSD::SetSignal in SALOME code so that FPE signals were disabled unless Salome is compiled in debug mode and the environment variable DISABLE_FPE is not set to 1.
In other words, in the optimized mode FPE are not raised, as it has been in earlier versions, and in the debug mode the code decides to raise or not FPE signals depending on the value of the variable DISABLE_FPE.

Splash logo modifications

Note that the processing of the splash screen window has been modified in SALOME:

1) A new class QtxSplash which presents the splash screen window similar to the Qt's QSplashScreen class (the latter does not suit all SALOME requirements) has been implemented.
2) The main difference from the previously used approach is that this widget can be shown even if the main application event loop has not yet been started, which allows displaying splash at the very first step.

3) Correspondingly SALOME_Session_Server has been improved to use this class instead of the previous one (InquireServersQThread.* files have been removed from the compilation process since they are no more needed).
4) Splash screen support has been implemented for the light SALOME configuration - the SUITApp package has been modified correspondingly.
5) Splash screen is now much more customizable than earlier (in both configurations - full and light). It is now possible to customize the following parameters of splash screen window:
    a. splash screen picture (file);
    b. progress bar color(s) (one color or two colors for gradient progress bar, vertical or horizontal type of gradient);
    c. splash screen info text (any text can be displayed in the splash window; there is also some predefined templates support: %A for application name, %V for application version, %L for license info and %C for copyright).
    d. splash screen text colors (one or two colors, second one is used for text shadow).

The parameters of splash screen are customizable via configuration files. Default values are already provided in the corresponding files: LightApp.xml for the light configuration and SalomeApp.xml for the full version of SALOME.

Support of LOD actor is added (useful in SMESH and VISU module for large models)

SALOME VTK viewer now has a new functionality - management of Level Of Details (LOD) rendering through the usage of the native VTK LOD functionality (see also http://www.vtk.org/doc/release/4.2/html/classvtkLODActor.html).
It allows the user to manage performance of rendering through the corresponding dialog. By default the dialog is disabled and the viewer behaves as before. To activate the functionality check "Enable" check box and click "Apply" or "Ok" buttons. In the dialog you can choose ether "Desired Update Rate" (by default 15 Frames Per Second) or "Still Update Rate" (by default 0.0001 FPS). The first parameter corresponds to the rendering when an action, for example: rotation, panning etc. is activated. The second parameter corresponds to the rate of restoring when an action is finished.

Other improvements

- Handling of references has been changed: by default, in the current version the user should remove invalid references to a deleted referenced object manually (through a popup). However each module can redefine such behavior and implement automatic deletion of references and removal of referenced objects.
- New improvement is realized a in a tabbed viewer. In the 3.2.0 version it is possible to rename and close viewers through popup menus.
- Display and Erase in a popup present only for objects which can be displayed in a current active viewer.
- Improvements in behavior of SALOME preferences.
- New improvements have been made in the object browser. Now double click on a reference object changes the selection to the original object. Also there is a possibility to work with references like with separate objects and introduce different logic comparing to the original object.
- Some improvements related to text the drawing performance and robustness are made into GLViewer package;
- Minor code amendments and document closing debug in STD, SUIT and Plot2d.
- If Mozilla browser is not installed, a special dialog gives the possibility to choose another browser as a default one.

GEOM module

Important remark: Currently there is special python interface geompy.py for providing access to GEOM functionality. This is right way of building python scripts. In the version 3.2.0 there are different examples in GEOM_SWIG which don't use this interface, but use direct calls of IDL functionality. They are not related to end user, but can be treated as non regression tests only. In future version they will be replaced by right end-user examples. Geom documentation provides correct examples inside.

New GEOM plugin for import of CATIA V5 files

OPEN CASCADE SAS is glad to present a new plugin for import of native CATIAV5 files into the GEOM module.

Important remark: The plugin is a wrapper for a library which converts CATIA files – this library is a commercial product protected by license. To receive an evaluation version of the converter please contact OPEN CASCADE SAS.

Repackaging of GEOM module

Repackaging of GEOM module packages has been performed in order to produce non-CORBA packages to simplify the engine architecture of this module. Now pure C++ layer with Open CASCADE technology functionality is separated from CORBA layer in special libraries.

New functionality available in IDL API

- The following methods were added into **GEOM_IBasicOperations** interface:

```
/*!
```

```
* Create a vector, corresponding to tangent to the given parameter on the given curve.
*  \param theRefCurve The referenced curve.
*  \param theParameter Value of parameter on the referenced
*                curve.This value should be have value
*  \between 0. and 1.. Value of 0. corresponds first parameter of
*                curve value 1. corresponds
*  \last parameter of curve.
*  \return New GEOM_Object, containing the created point.
*/

 GEOM_Object MakeTangentOnCurve (in GEOM_Object theRefCurve,
        in double theParameter);


/*!
*  Create a shell or solid passing through set of sections.Sections
*  should be wires,edges or vertices.
*  \param theSeqSections - set of specified sections.
*  \param theModeSolid - mode defining building solid or shell
*  \param thePreci - precision 3D used for smoothing by default 1.e-6
*  \param theRuled - mode defining type of the result surfaces
*             (ruled or smoothed).
*  \return New GEOM_Object, containing the created shell or solid.
*/

 GEOM_Object MakeThruSections(in ListOfGO theSeqSections,
        in boolean theModeSolid,
        in double thePreci,
        in boolean theRuled);

/*!
*  Create a shape by extrusion of the profile shape along
*  the path shape. The path shape can be a wire or an edge.
*  the several profiles can be specified in the several
*  locations of path.
*  \param theSeqBases - list of  Bases shape to be extruded.
*  \param theLocations - list of locations on the path corresponding
*                specified list of the Bases shapes.
*                Number of locations
*                should be equal to number of bases or
*                list of locations can be empty.
*  \param thePath - Path shape to extrude the base shape along it.
*  \param theWithContact - the mode defining that
*                 the section is translated to be in
*                 contact with the spine.
*  \param - WithCorrection - defining that the section is rotated
*                    to be orthogonal to the spine
*                    tangent in the correspondent point
*  \return New GEOM_Object, containing the created pipe.
*/
```

```
GEOM_Object MakePipeWithDifferentSections (in ListOfGO theSeqBases,
       in ListOfGO theLocations,
       in GEOM_Object thePath,
       in boolean theWithContact ,
       in boolean theWithCorrection );

/*!
   * Create a tangent plane to specified face in the point with specified parameters.
   * Values of parameters should be between 0. and 1.0
   * \param theFace - face for which tangent plane should be built.
   * \param theParameterU - value of parameter by U
   * \param theParameterV - value of parameter V
   * \param theTrimSize - defines sizes of created face
   * \return New GEOM_Object, containing the face built on tangent plane.
   */
     GEOM_Object MakeTangentPlaneOnFace(in GEOM_Object theFace,
                 in double theParameterU,
                 in double theParameterV,
                 in double theTrimSize);
```

Additional methods in GeometryGUI_SWIG interface

The following functions were added into the GEOM SWIG interface (functions that can be called from the python console or any python GUI module).

```
/* Set display deflection for a shape presentation */

void setDeflection(const char* theEntry, float theDeflection);

/* Erases graphical object. If allWindows is true, then it will be erased from all windows,
otherwise only from current */

void eraseGO(const char* Entry, bool allWindows);
```

Other GEOM improvements

- Like the « GetShapesOnXXX » functions specialized to build geometric group, GetShapesOnQuadrangle is a new function that helps to build 2D geometric groups.
- GUI access has been added in GEOM to check and correct compound of blocks functionality. It can be accessed via Measures -> Check Compound of Blocks.

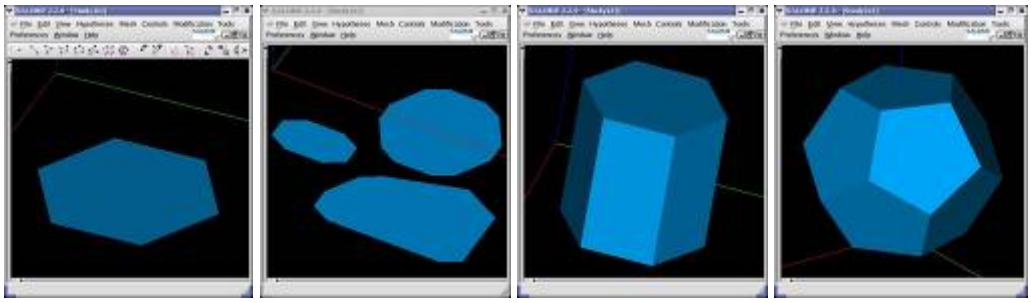
MESH module

*Important remark: Currently there is special python interface smesh.py for providing access to SMESH functionality. This is right way of building python scripts. In the version 3.2.0 there are different examples in SMESH_SWIG which don't use this interface, but use direct calls of IDL functionality. They are not related to end user, but can be treated as non regression tests only. In future version they will be replaced by right end-user examples. Please refer to "ex21_lamp.py" as a correct example, or other "ex_***.py" files.*

Polygon and polyhedral support

The SMESH module has been updated to add the possibility for creation of polygons and polyhedral elements

Operations on mesh now support the creation of polyhedral elements (sewing, pattern mapping, extrusion, revolution etc)



General redesign of GUI for mesh creation

In order to reflect dependencies between algorithms and between the algorithm and hypotheses, the dialogue for creation/edition of meshes/sub meshes has been improved.

The dialogue contains 3 tabs, each defining hypotheses of a certain dimension. Each tab has

- a list of available algorithms (combobox);
- two lists of existing hypotheses (comboboxes): for main and additional hypotheses;
- two "Create hypothesis" buttons, which, when pressed, show a list of available hypotheses;
- Two "Edit hypothesis" buttons invoking a dialogue for editing of the hypothesis selected in the list of existing hypotheses.

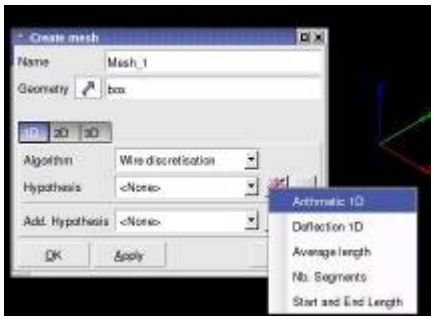
The algorithm - algorithm dependencies are reflected as follows.
When an available algorithm type is selected, the lists of algorithms in the other tabs are updated:

- lists of available algorithms change to contain only the algorithms compatible with the selected one;
- if a pre-selected algorithm is incompatible with the selected one, it is deselected;
- if there is only one compatible algorithm in the list, it is selected;
- if no algorithms are selected in a tab, the other tabs contain all available algorithms.

The algorithm - hypotheses dependencies are reflected similarly.
When an algorithm is selected, the lists of available and existing hypotheses in each tab are updated to contain only hypotheses compatible with the algorithm of the tab, selected by the user or automatically:

- if there exists only one compatible hypothesis in the list, and it is not optional, it is selected;
- if no algorithms are selected, then the lists of hypotheses are empty.

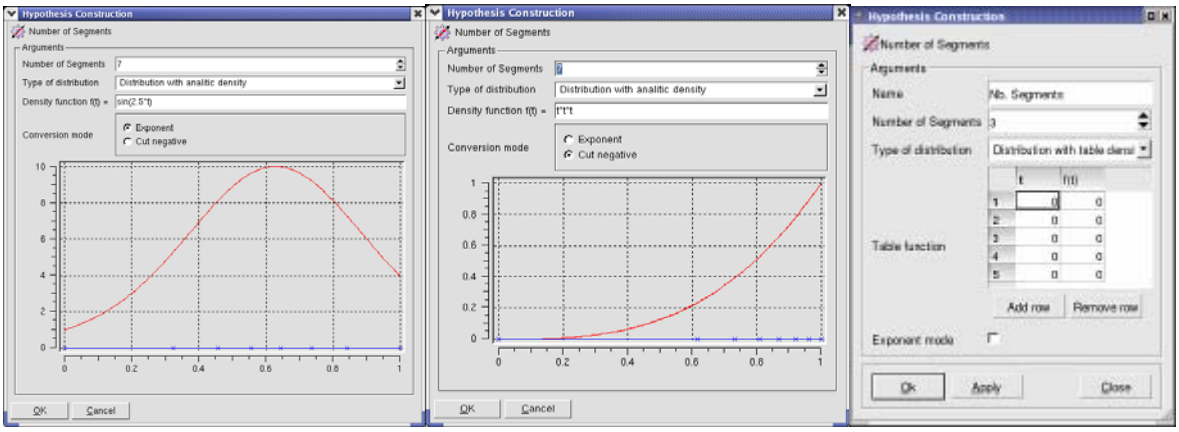
The order of tabs has been changed: the tab for 3D Hypotheses comes first. As currently, for most 3D algorithms (except GHS3D), there is only one compatible 2D and 1D algorithm, when a 3D algorithm is selected, all other algorithms are selected automatically.
The tab of highest accessible dimension always contains all types of algorithms of this dimension.
All GUI controls providing selection from a list are disabled when the list is empty.



Improvement of 1D hypothesis "Number of segments"

The 1D hypothesis "Number of segments" is now more generalized, providing a powerful functionality for creation of points on edges using the following distributions:

- Regular distribution of points
- Non regular using scale factor (arithmetic progression)
- Table distribution
- Analytical formula for point distribution
- Preview for tables and analytical presentation - draw of function and distribution of points
- Numerical algorithm for points calculation based on density function is improved to provide robust results



Quadratic Elements Implementation in the SMESH module

Quadratic mesh elements are frequently used in mesh algorithms. Implementation of mapping of MED quadratic elements gives the possibility to use more advanced features of meshers and solvers.
For visualization of quadratic mesh elements we will use native VTK mesh elements.
Implementation of quadratic mesh elements in the SMESH module has caused many changes of data structure, algorithms, visualization and interfaces.

1. Changes in SMDS_MeshElement class
   To distinguish quadratic elements from the regular ones, virtual method IsQuadratic() can be defined in SMDS_MeshElement class and redefined in classes representing quadratic elements. In order to be able to differ corner nodes of elements from medium nodes located on edges, a virtual method IsMediumNode(const SMDS_MeshNode* node) can be defined in SMDS_MeshElement class:KERNEL module improvements
1. SMDS_Mesh interface for quadratic volumes has been created to add methods for creation of volumes with 10, 13, 15 or 20 nodes. For example for addition of a tetrahedron with 10 nodes:
2. To maintain quadratic volumes, a new class SMDS_QuadraticVolumeOfNodes, inheriting SMDS_MeshVolume has been implemented.

3. SMDS_Mesh interface for quadratic faces has been created to add methods for addition of a triangle with 6 nodes and a quadrangle with 8 nodes.
4. New class for quadratic faces SMDS_QuadraticFaceOfNodes, inheriting SMDS_MeshFace has been implemented.
5. SMDS_MeshEdge* AddEdgeWithID and SMDS_MeshEdge* AddEdge methods have been added to SMDS_Mesh interface for creation of quadratic edges.
6. Class SMDS_QuadraticEdge inheriting from SMDS_MeshEdge represents a quadratic edge

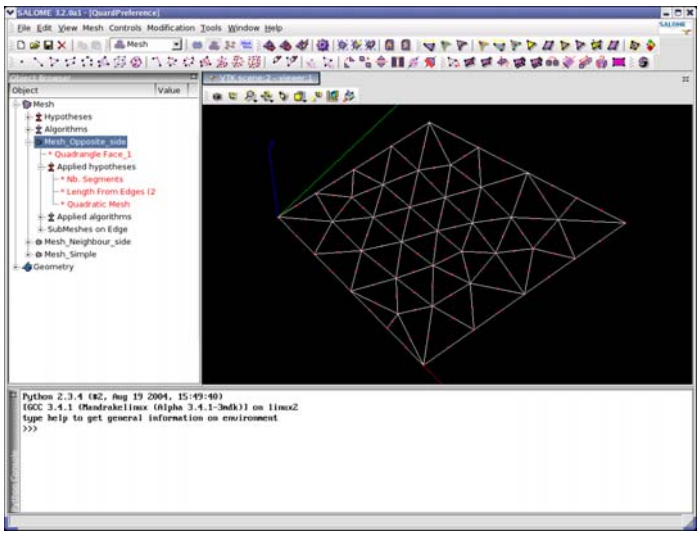
Such operations on meshes with quadratic elements, as sewing and nodes merging are also supported through implementation of support of quadratic volumes by the SMDS_VolumeTool class and SMESH_MeshEditor.
In order to force generation of a quadratic mesh, an auxiliary "Quadratic Edge" 1D hypothesis has been added. If it is assigned, a geometric edge is meshed with quadratic 1D elements.
If a 2D mesher sees that all boundary edges are quadratic, it generates quadratic faces, else it generates linear faces using medium nodes as if they were vertex.
The 3D mesher generates quadratic volumes only if all boundary faces are quadratic, else it fails.

The meshers implemented in SALOME, i.e. StdMeshers_Regular_1D, StdMeshers_Quadrangle_2D, StdMeshers_Penta_3D and StdMeshers_Hexa_3D, have been modified to allow quadratic mesh generation.
The external meshers not supporting quadratic elements (MEFISTO) will be fed with linear input elements by ignoring medium nodes of quadratic boundary elements. The output of such external meshers will be modified by addition of medium nodes on edges of generated linear elements; for 2D elements, added medium nodes will lay on the meshed geometric surface; for 3D elements, they will be added in the middle of an edge.

Advanced mesh info has been updated to show the number of quadratic elements

Special functions allowing converting quadratic elements to usual first order elements and visa versa have been added in SMESH module

The NETGEN external mesher already supports quadratic elements.



Quadratic triangles

Visualization of quadratic meshes in the VTK viewer has been also improved by introducing a true representation of the quadratic mesh elements in the shading mode by means of reconstructing sides of objects with the help of quadratic surfaces and removal of internal triangulation of quadratic mesh elements in the wireframe mode.

Optimization of SMESH module performance

The design and functionalities of SMESH module have been thoroughly analyzed to increase their performance.

Mesh structure is not copied from SMESH CORBA server to the client side (GUI) when they are running on one workstation, and SALOME Container runs in embedded mode. Instead, a separate SMESH_Client class from SMESH_Object and other GUI classes are implemented. SMESH_Client checks if server and client processes have identical PID and run on the same workstation. If this is the case, mesh data structure is not updated by passing the SMESH script with mesh creation and modification history through CORBA calls. Instead, a C++ pointer to the corresponding SMESH_Mesh instance is passed directly for future visualization.

Interactive selection mechanisms have been optimized with a significant performance progress since SALOME 3.2.0 taking into account the optimization in the framework of Gauss Viewer project. Multiple viewer updates have not been found during a local selection (remove elements operation) in the current development version. The current results of performance tests seem to be acceptable even for large meshes (~1.5M elements).

The implementation of SMESH_Client class has improved not only performance but also memory usage in embedded mode. This gives ~40% less memory usage in case of 200-thousand elements mesh for instance and also ~50% faster first display of the same mesh.

NETGEN 2D-3D plugin

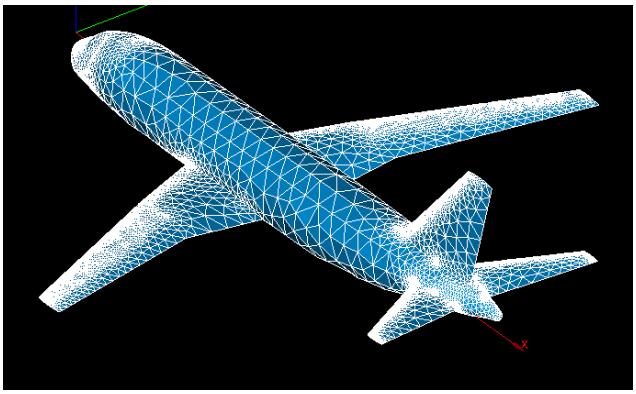
The study on NETGENPLUGIN porting resulted in the adoption of a new version of Netgen 4.5 (before that Netgen 4.3 had been used).

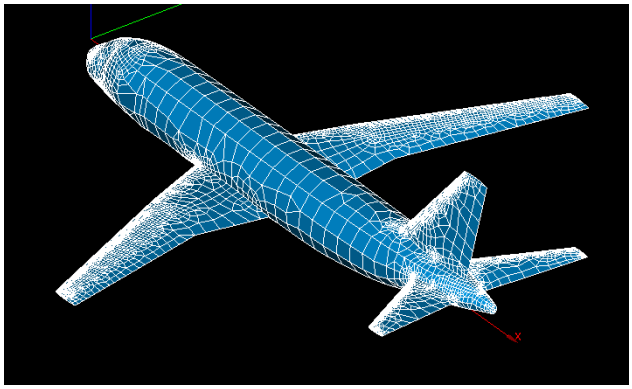
It includes

- Porting NETGENPLUGIN module to Netgen ver.4.5.
- Adding two new independent algorithms to generate 2D and 3D meshes directly from OCT shape.
- Adding two new hypotheses for new algorithms.
- Adding a GUI library providing a dialog to edit hypotheses

The following minor modifications have been introduced into SMESH module:

- Specific icon files for Netgen plugin to be displayed in the object browser have been into the resources of SMESH module.
- SMESH_SWIG/smesh.py file has been modified to support new Netgen algorithms in this python module.
- Two new python test scripts: SMESH_mechanic_netgen.py and SMESH_fixation_netgen.py has been added into SMESH_SWIG directory. The first generates a mixed quadrangle/triangle 2D mesh. The second generates a tetrahedral 3D mesh.





"Flight.brep" model which failed with default parameters on MEFISTO algorithm is now computed perfectly

Plugin modifications

To be able to import the module of SMESH plugin into the script produced by DumpPython functionality, the following method have been added into GenericHypothesisCreator_i class (SMESH_SRC/src/SMESH_I/SMESH_Hypothesis_i.hxx), which is a root of classes creating hypotheses in plugins:

```
class GenericHypothesisCreator_i
{
public:
  ...
  // return the name of IDL module
  virtual std::string GetModuleName() = 0;
};
```

This method returns the name of python module defining classes of hypotheses created by a plugin; this module is generated from IDL file of the plugin.

The method is defined in, for example, NETGENPLUGIN as follows (NETGENPLUGIN_SRC/src/NETGENPlugin/NETGENPlugin_i.cxx)

```
template <class T> class NETGENPlugin_Creator_i:public
HypothesisCreator_i<T>
{
  // as we have 'module NETGENPlugin' in NETGENPlugin_Algorithm.idl
  virtual std::string GetModuleName() { return "NETGENPlugin"; }
```

```
};
```

As a result, DumpPython script contains lines importing modules of plugins which have created some hypotheses:

```
def RebuildData(theStudy):
    ...
  import StdMeshers
  import NETGENPlugin
```

Update of MEFISTO algorithm

StdMeshers package has been updated by the latest version of MEFISTO algorithm. In some case it works better, but sometimes it produces bad meshes.

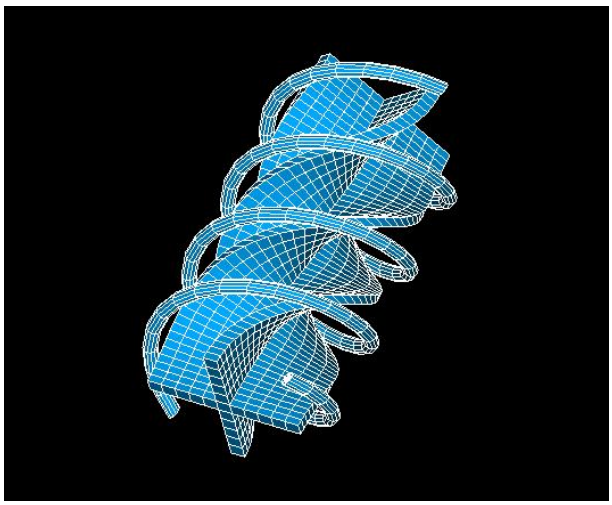
UNV import-export improvements

From now on, the UNV interface supports reading-writing of groups.

New IDL functionality

New functions have been added to SMESH. They provide the possibility to work with mesh without GEOM data. For more details please refer to IDL documentation.

```
double_array GetNodeXYZ(in long id);
long_array GetNodeInverseElements(in long id);
long GetShapeID(in long id);
long GetElemNbNodes(in long id);
long GetElemNode(in long id, in long index);
boolean IsMediumNode(in long ide, in long idn);
long ElemNbEdges(in long id);
long ElemNbFaces(in long id);
boolean IsPoly(in long id);
boolean IsQuadratic(in long id);
double BaryCenter(in long id);
```



Mesh generated by a script without geometry
(SMESH_AdvancedEditor.py example script)
The idea is taken from GMESH web site.

Created by Open CASCADE SAS for CEA/EDF in the frame of the PAL/SALOME contract

Other minor improvements

- Functionality for mesh extrusion was improved and currently includes parameters for automatic Sewing calling after the extrusion of 2D mesh and an option to switch on/off the building of 2D skin mesh.
- Export to the med dialog box now allows controlling the creation of default groups of elements in med files.
- Portability with gfortran of MEFISTO algorithm has been applied
- In addition the functionality of SMESH module accessible from GUI was revised to be coherent with all available functions of the TUI mode. As a result, now the GUI of TriangleToQuadrangle and QuadrangleToTriangles operations includes additional options giving a choice from the two possible variants, depending on the quality (for example, aspect ratio) and the maximum angle allowed between two triangles to unite them in a quadrangle.
- SMESH now provides computation of volume mesh by NETGEN without hypothesis by default
- Introduction of new SMESH 3D Control: volume of the tetra/hexa.
- New feature is available - automatic 1D mesh creation. Automatic length hypothesis provides a possibility to generate default mesh based on minimal length of the edge and bounding box of model.
- A new color attribute has been introduced in the notion of mesh groups. This attribute is being written into MED file as a family attribute and can be used in computation to assign boundary conditions.

MED module

Polyhedral and polygon elements support.

- MED wrapper class in KERNEL has been updated to support reading and writing of polyhedral elements.
- IDL interfaces as well as MEDMEM have been modified in order to support this new feature.

Other improvements

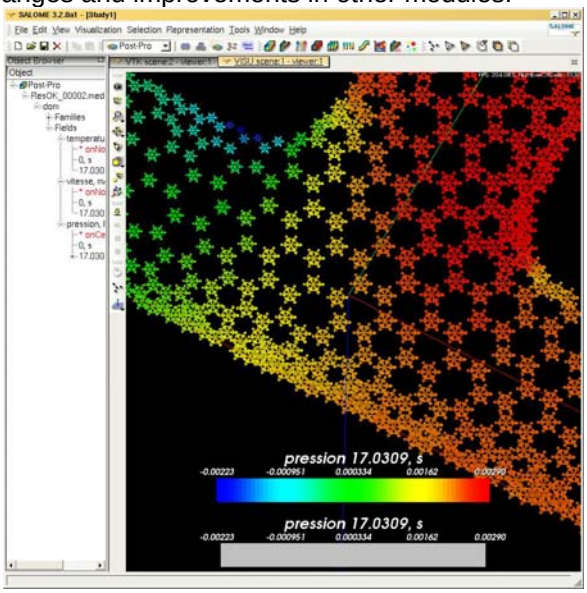
- Compilation of MED component without connection to KERNEL. It's done to have a standalone MED component (mainly MEDMEM) for other projects. It can be done using options '--without-kernel' or '--with-kernel=no' for build_configure steps. It's not necessary to call configure with this options.
- From now on the MED module can save its data within the SALOME study. This allows external modules to write down their MEDMEM fields published in the study
- As a result of Gauss Viewer implementation now binary libraries of boost are used; gauss points and MED profiles are supported in MEDWrapper.
- Thread-safe support in MEDWrapper, some small modifications of interface of some MEDWrapper classes.
- Inconstant behaviour between V2.2.x and V3.2.0 has been fixed.
- Several bugs found has been fixed: set field value type at creation; fix in typemap for vectors passed by value for SWIG 1.3.21.
- The Med File drivers of the Med Memory support the V2.1 as well as the V2.2 versions of the Med File layer. The requirement of the Salome platform is only Med File V2.2, the V2.1 version is embedded in the Med Memory.
- Using the Med file (V2.1 and V2.2) and GIBI drivers; fields laying on a partial support;
- Fields defined on cells mesh with multiple gauss points, may be mounted in memory and treated.
  With all those new functionalities, most of the previous work based on previous releases of the Med Memory should work; but minor changes should be done for the get/set field class methods:
  - the getValue() and the setValue(T *) methods take no MED_EN::medModeSwitch parameter;

- the getValueI (resp. setValueI) should be replaced by getRow(int) (resp. setRow(int ,T*) if the field is in full interlacing mode (using the method getInterlacingType() of the classe FIELD_). If the field is stored in no interlacing mode getValueIJ (resp. setValueIJ) should be replaced by getColumn(int) (resp. setColumn(int ,T*)).

- Intensive debugging was carried throughout the entire Med Memory C++ Layer, especially on the major user's C++ classes (such as MED, MESH, SUPPORT and FIELD); the C++ drivers classes on those major classes. Especially the Med File and the GIBI drivers are read/write ones. The VTK drivers are only for the writing; and finally the PORFLOW drivers may only be used for the MESH class in the reading mode.
- The Med Client layer of the Med Memory has been improved in a full Server/Client configuration.

VISU module

Salome now supports presentation of gauss points in special dedicated viewer. This implementation necessitated the following changes and improvements in other modules.

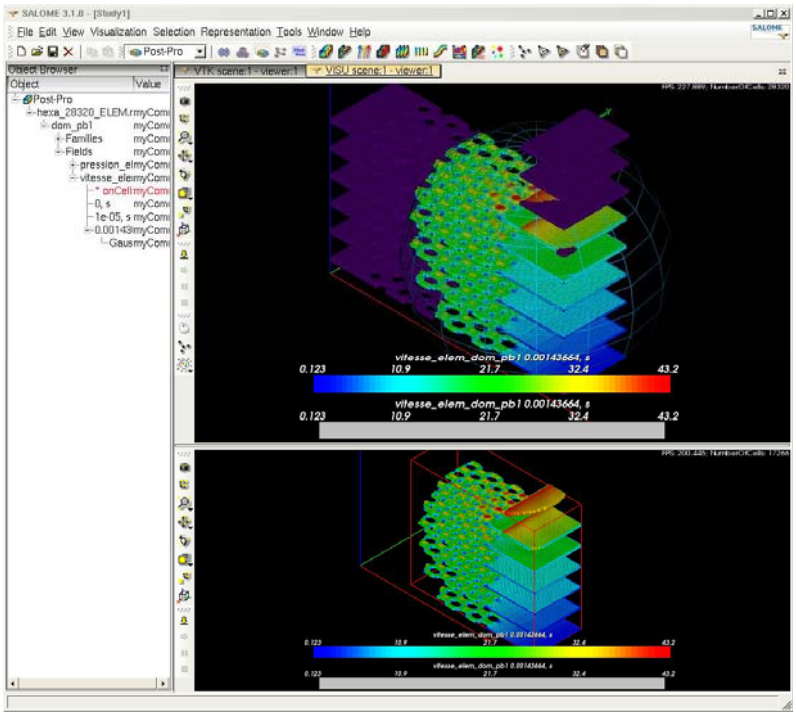


Improvements of the SALOME VTK viewer (SVTK).

- Scaling and Graduated Axis functionality is generalized and moved from VISU to the GUI module.
- SVTK_ViewWindow can contain more than one instance of the vtkRenderWindow which is a considerable improvement of the layout management.
- Native VTK events are now fully supported which means that now it is possible to use vtk3DWidget subclasses to enrich visualization pipeline.
- It is possible to customize selection and pre-selection functionalities (every presentation can override the default selection and pre-selection functionality).
- New selection modes can be introduced in a simpler way because now every presentation can introduce and process its own selection mode.
- Visualization of the quadratic mesh elements is supported.
- Interaction with Space Mouse (external device of virtual reality) is supported.
- The style of the user interaction became customizable.

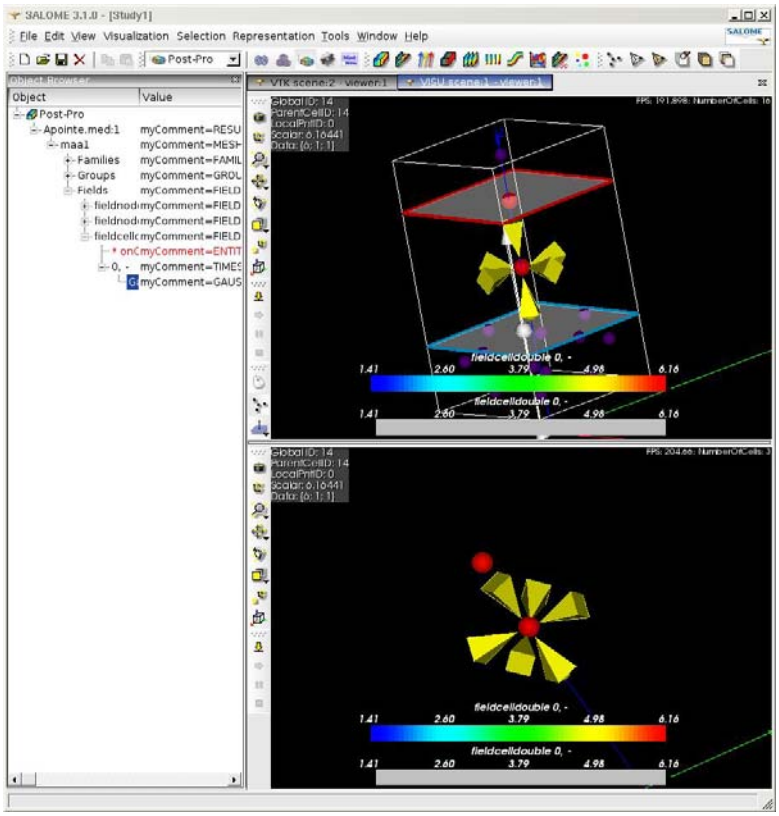
Improvements of the SALOME MEDWrapper package.

- MED PROFILE entity is fully supported (it is possible to get know whether the assigned data is defined on the whole MED ENTITY or on the specified sub-set of mesh elements only).
- MED GAUSS entity is fully supported (it is possible to read and calculate the absolute 3D space position of Gauss Points of the calculated data).
- Thread-safety for the package is implemented.



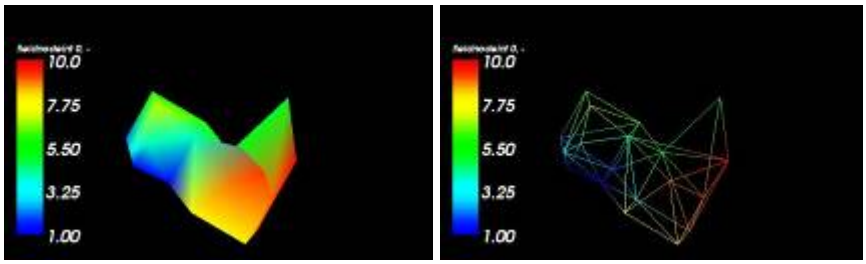
Modification of the SALOME VISU module.

- A new type of the 3D presentation – Gauss Points is implemented.
- OpenGL extensions now provide fast rendering for Gauss Points presentations.
- Background loading of MED files is implemented.
- The process of the loading of the MED files became customizable (it is possible to choose what MED entities will be loaded).
- Support for visualization quadratic mesh elements is provided.
- Full support for multi-component MED fields is provided.
- A new viewer, sub type of the SVTK, is implemented. Its users now can customize:
  - layout management (the viewer provided two views instead of a single view);
  - style of the user interaction (the user can operate in the viewer using the keyboard).
- AVI recording functionality for the sub-type of the SVTK viewer is implemented.



Polygon and polyhedral support

- VISU module can visualize polyhedral elements and data field on them
- VTK visualization classes have been updated to support the display of polygons/polyhedrons



Visualization parameters persistence in study and Dump python for GUI in VISU module

Now in Salome VISU module the visual parameters of a saved Salome document, such as the number of opened viewers, types of opened viewers, viewer parameters (orientation, zoom), color, transparency, the type of presentation of displayed objects and the type of displayed entities are also recorded in the document. When the saved document is opened again its visual parameters are also restored and the objects are rebuilt, which means that the session is restored exactly in the same way as it has been saved. This functionality is also supported for all other Salome modules; however, there the objects are not rebuilt automatically at loading.

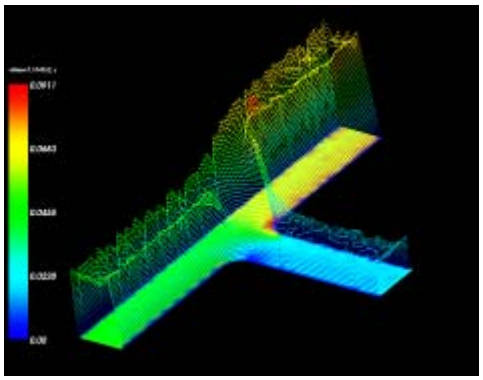
This functionality also works at the generation of Python scripts (Dump Python functionality).

A new check box added in preferences must be checked on if the user wants to use the proposed functionality.
This new functionality has some limitations; obviously, the visual parameters of the documents created with the previous versions of Salome can't be restored. Due to the fact that all objects displayed in the viewers can belong to the different modules all modules will be loaded at the opening of a document. Thus if the viewer is used by more than one module to display the objects the viewer parameters (orientation and zoom) will correspond to the parameters of the last loaded module that uses that viewer.

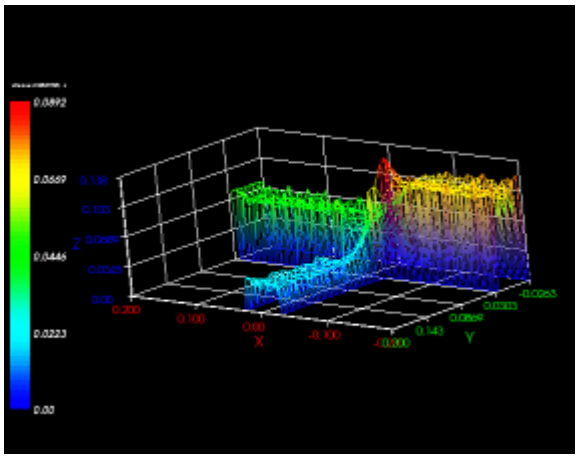
- Addition of preferences in VISU module: now general preferences contain preferences for VISU presentations (colors, type of presentation etc)
- Toolbar buttons to import/export MED files have been added in the module
- Structured mesh elements in VISU are now supported!
- Generation of AVI animations: the user can also choose generation of animation during generation of pictures (the availability of this functionality depends on existence of mpeg tools in Linux)

Other major improvements in the VISU module

- Clipping plane. Like the SMESH module, VISU now has “clipping plane presentation”. It includes classical clipping and also i-j-k in case of a structured mesh.
- Merging scalar bars. VISU allows to have global scalar bar for different presentations now
- 2D fields displayed in 3D. New kind of presentation has been introduced – it builds 3D presentation based on assigned fields for mesh



- Translation of presentations. The user can translate presentations in 3D viewer in order to see different results simultaneously in one space
- New animation mode based on translated presentations
- Graduated bounding boxes. Graduated rules are accessible through a menu now – it allows to see the dimension of presentations



Other minor VISU improvements

- Multiple delete operation
- Now set axis scale is available from TUI
- Currently shading for visualization in VTK viewer can be controlled through preferences
- A new improvement was introduced for creation on cut lines and curves based on them
- There are new features: preview of order of cut lines using the marker in the viewer and also a possibility to reverse them.
- It's possible now to generate Plot3D presentation based on single Cut Plane.

SUPERVISOR module

Improvements for Supervision messages classification has been carried out.
The IDL of SUPERVISION module has been modified:

- A new method has been added into FNode interface

```
/*! Method to define if %FNode gets from a component, which is implemented in
    C++ or Python.  Returns True if it's a C++ component node
*/
  boolean IsCimpl() ;
```

- The signature of method FNode from Graph interface has been changed from

```
/*! Creates a Factory Node in a Graph.
*/
SUPERV::FNode FNode( in string aComponentName ,
                     in string anInterfaceName ,
                     in SALOME_ModuleCatalog::Service aService ) ;
```

to

```
/*! Creates a Factory Node in a Graph.
*/
SUPERV::FNode FNode( in string aComponentName ,
                     in string anInterfaceName ,
                     in SALOME_ModuleCatalog::Service aService ,
                     in boolean isCimpl ) ;
```

General documentation improvements

Renovated and significantly improved documentation in the part of modules was added providing examples and TUI command interface description.

In general please have a look at the documentation of installation of/running SALOME, its general desktop organization, KERNEL operations and end user actions for SMESH and GEOM modules with the corresponding examples of python scripts.

These help items are available via the SALOME Help menu and provide access to modules user help documentation.
Please note that by default compilation procedure does not generate sources API documentation for most modules (except KERNEL). To build this documentation please call “make dev_docs” step before call of “make install”.

New samples

New sample LIGHT has been developed by Open CASADE SAS. It is called LIGHT_SRC and placed in EXAMPLES CVS. This module demonstrates how to write a simple CORBA-light component.

New sample developed by CEA has been adopted for SALOME 3.2.0. It is called CALCULATOR_SRC and it demonstrates how to write a C++ Salome Module which is interfaced with MED Memory, using MedClients classes. Associated documentation in sxw and pdf format can be found in the /doc directory.

New sample SIERPINSKY developed by Open CASCADE SAS is adopted for SALOME 3.2.0. It is called SIERPINSKY_SRC and RANDOMIZER_SRC and it demonstrates how to write a simple C++ and Python Salome Module which is interfaced with both VTK viewer and MED component. The example implements a simple interface to calculate Sierpinsky fields. Please read README in SIERPINSKY_SRC about the configuration and usage of this example
It can be retrieved from CVS EXAMPLES:
:pserver:<username>@cvs.opencascade.com:/home/server/cvs/EXAMPLES.

Module name is SIERPINSKY_SRC and RANDOMIZER_SRC.

Supported Linux distributions and pre-requisites

SALOME 3.2.0 supports Mandrake 10.1, Debian Sarge, Mandriva 2006, RedHat 8.0, 9.0, RedHat Enterprise 4, Saintific Linux 3.1 and Mandriva 64 bit. Please note that SALOME is not certified in this latter OS.

Note: OPEN CASCADE TECHNOLOGY version has been changed to OCT 6.1 official version (can be downloaded from www.opencascade.org site).

New NETGEN 4.5 versions comes with installation procedure and new latest doxygen version 1.4.6 which support generation of documentation from python.

New prerequisite comes with installation procedure – docutiles 0.3.7. Necessary for generation of documentation in KERNEL.

HappyDoc is removed from third party products

SALOME 3.2.0 version has been mainly tested with the following pre-requisite list on Mandrake 10.1 platform.

Following set of prerequisites are valid for SALOME 3.2.0 version. Please note that we try to use as much as possible native products.

	Mandriva 2006	Debian Sarge	Mandrake 10.1	RedHat Enterprise 4	RedHat 8	RedHat 9	RedHat Scientific 3.0.5
gcc	4.0.1	3.3.5	3.4.1	3.4.1	3.2	3.2	3.2
tcltk	8.4.5	8.4.5	8.4.5	8.4.5	8.0	8.0	8.0
Python	2.4.1	2.3.5	2.3.4	2.3.4	2.3.4	2.3.4	2.3.4
Qt&msg2qm	3.3.4	3.3.4	3.3.3	3.3.3	3.3.3	3.3.3	3.3.3
Sip	4.2.1	4.1	4.1	4.1	4.1	4.1	4.1
PyQt	3.14.1	3.13	3.13	3.13	3.13	3.13	3.13
Boost	1.32.0	1.31.0	1.31.0	1.31.0	1.31.0	1.31.0	1.31.0
Swig	1.3.24	1.3.24	1.3.24	1.3.24	1.3.24	1.3.24	1.3.24
OpenCascade Technology	6.1	6.1	6.1	6.1	6.1	6.1	6.1
Qwt	4.2.0	4.2.0	4.2.0	4.2.0	0.4.1	0.4.1	0.4.1
OmniORB	4.0.6	4.0.5	4.0.5	4.0.5	4.0.5	4.0.5	4.0.5
Hdf	5-1.6.4	5-1.6.3	5-1.6.3	5-1.6.3	5-1.6.3	5-1.6.3	5-1.6.3
Med	2.2.3	2.2.3	2.2.3	2.2.3	2.2.3	2.2.3	2.2.3
Vtk	4.2.6	4.2.6	4.2.6	4.2.6	4.2.2	4.2.2	4.2.2
Numeric	23.7	23.7	23.7	23.7	22.0	22.0	22.0
Graphviz	2.2.1	2.2.1	2.2.1	2.2.1	1.9	1.9	1.9
Doxygen	1.4.6	1.4.6	1.4.6	1.4.6	1.4.6	1.4.6	1.4.6
NETGEN	4.5	4.5	4.5	4.5	4.5	4.5	4.5
docutils	0.3.7	0.3.7	0.3.7	0.3.7	0.3.7	0.3.7	0.3.7

However, other versions of pre-requisites may also work.

How to install and build SALOME

- Please follow README file from Installation Wizard for processing correctly installation of SALOME and all prerequisites
- If you would like to compile SALOME from scratch, please use build.csh script delivered with Installation Wizard. Call “build.csh –h” to see all parameters of this script.
  - *Important remark*: on RedHat 8 with the native automake-autoconf tools, sources of KERNEL from CVS can not be compiled. As a workaround there is specially prepared sources of KERNEL in Installation Wizard (after “make dist” step from Mandrake 10.1). They can be compiled with old prerequisites, but user must not call “build_configure” step. To compile he must call “configure”, “make”, “make install” as usual. Because of this please don’t use “build.csh” with “–b” option for KERNEL, because this option forces build_configure step. Call of “build.csh –i –o” process compilation and installation on RedHat8 correctly.

How to get the version and pre-requisites

The SALOME 3.2.0 pre-compiled binaries for Mandrake 10.1, Debian Sarge, Mandriva 2006 and RedHat 8.0 can be retrieved from the PAL/SALOME FTP site (ftp://www.opencascade.com).
There are sources of modules inside, and user can build sources from scratch using “build.sh” script coming with installation procedure.
Alternatively, SALOME modules can be downloaded from the following CVS repositories:

- KERNEL module: :pserver:<username>@cvs.opencascade.com:/home/server/cvs/KERNEL
- GUI module: pserver:<username>@cvs.opencascade.com:/home/server/cvs/GUI

- GEOM module: pserver:<username>@cvs.opencascade.com:/home/server/cvs/GEOM
- MESH module: pserver:<username>@cvs.opencascade.com:/home/server/cvs/SMESH
- SUPERVISOR module: pserver:<username>@cvs.opencascade.com:/home/server/cvs/SUPERV
- VISU module: pserver:<username>@cvs.opencascade.com:/home/server/cvs/VISU
- MED module: pserver:<username>@cvs.opencascade.com:/home/server/cvs/MED
- NETGEN plugin module: pserver:<username>@cvs.opencascade.com:/home/server/cvs/NETGENPLUGIN
- SAMPLES: pserver:<username>@cvs.opencascade.com:/home/server/cvs/EXAMPLES

IMPORTANT! Source files for version 3.2.0 are available in CVS via tag **V3_2_0**

The patch on NETGEN is placed inside NETGENPLUGIN sources. During the compilation of a plug-in, the patch is applied automatically to the standard NETGEN installation.

All other pre-requisites shall be obtained either from your Linux distribution (*please be sure to use a compatible version*) or from the distributors of these pre-requisites (*www.trolltech.com for QT for example*).

Known problems and limitations

- On Mandriva 2006 salome_test fails due bug in omniOrb 4.0.5. Bug concerns precision of conversion of double from python to C++ stubs. This bug is fixed in omni 4.0.7.
- Due to a bug with changes of tolerance in OCT 6.1, GUI test scenario PROD 04 can not be played in TUI mode without workaround. Workaround is to call the same Boolean operation cut twice
- During the compilation of OCT 6.1 by makefiles on a station with NVIDIA video card you may experience problems because the installation procedure of NVIDIA video driver removes library libGL.so included in package libMesaGL from directory /usr/X11R6/lib and places this library libGL.so in directory /usr/lib. However, libtool expects to find the library in directory /usr/X11R6/lib, which causes compilation crash (See /usr/X11R6/lib/libGLU.la). We suggest making links:
  “ln -s /usr/lib/libGL.so /usr/X11R6/lib/libGL.so ln -s /usr/lib/libGL.la /usr/X11R6/lib/libGL.la”
- Partition algorithm has some problems on Debian Sarge. Due to this “Check geom” fails.
- OCT 6.1 has problems in debug mode on Debian Sarge
- VISU module does not support timestamps defined on the same field but on different meshes
- In the current implementation of “Save VISU” state operation the parameters of Gauss view Partition mode are not stored. If a window has been partitioned and saved, it will be restored as non-partitioned. The same concerns background color.
- Second run of ex00_all.py on Debian lead to crash due to noted problems in partition algorithm
- Fails of display of some presentation on quadratic elements in VISU (cannot create animation for IsoLines, CutPlanes etc.) is inside of the VTK. Currently used version of the VTK library (4.2.6) can not properly process the quadratic mesh elements (only ScalarMap and DeformedShape can be created only) that is presented in the MED file. Unfortunately it is impossible to replace or overload the VTK functionality outside of the library. This problem will be fixed automatically when we port the SALOME platform on the VTK 5.0 or higher version). This concerns Gauss viewer on quadratic elements. On some files with quadratic elements it is impossible to build gauss presentation.
- Problem of TestVisu20.xml failure in Supervisor still exists on 3.2.0 version
- Step-by-step execution in SUPERVISOR on some graphs fails. This functionality is only a prototype and was not finished completely
- Due to VTK 4.4 limitation, display of numbers of nodes in SMESH module have problems (some numbers disappears from viewer)
- VTK presentation in GEOM was not completely finished and has problems with performance and memory usage. It desirable to use OCT viewer in GEOM module.
- MEFISTO algorithm fails on some cases
- Due to some opened bugs, some non regression TUI tests fail:
  - MED component

    - SMESH module
    - GEOM module, partition problems
- End user documentation for Supervisor module was updated only in parts of screenshots.
- Results of some TUI non-regression testing are different on slow and fast computers. This is due to using in some test cases the functionality of GEOM from Supervisor in parallel nodes. Due to the fact that Open CASCADE Technology (OCT) does not thread safe, in some hardware configurations there is parallel conflict access to some data and such supervisor graphs fail. This problem will be fixed in future. At this moment the workaround is usage of GEOM nodes subsequently.